RCI-TR-005
LO-726925 9950-536

Final Report, Analysis Of DSN Software Anomalies

27 February 1981

Prepared By:

D.D. Galorath, H. Hecht*,
M. Hecht* and D.J. Reifer

Prepared For:

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91103

*SoHaR, Inc.

Reifer Consultants, Inc.

2733 Pacific Coast Highway, Suite 203 • Torrance, California 90505

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd.)

TABLE OF CONTENTS (cont'd.)

SUMMARY

The purpose of this study was to obtain a categorized data base of software errors which have been discovered during the various stages of development and operational use of the Deep Space Network DSN/Mark 3 System.

To achieve this purpose the Reifer Consultants Inc. (RCI) study team identified several existing error classification schemes (taxonomies), prepared a detailed annotated bibliography of the error taxonomy literature, and produced a new classification scheme which was tuned to the DSN anomaly reporting system and encapsulated the work of others. Based upon the DSN/RCI error taxonomy, error data on approximately 1000 reported DSN/Mark 3 anomalies was analyzed, interpreted and classified. Next, error data was summarized and histograms were produced highlighting key tendencies.

Finally, Reifer Consultants Incorporated recommended further statistical analysis of the software error data base so that trends and tendencies could be analyzed and the data could be consistently and meaningfully interpreted. Recommendations for other studies were also made as a result of this effort's findings.

ACKNOWLEDGEMENTS

The work reported here was performed by Reifer Consultants, Incorporated for the Jet Propulsion Laboratory under Purchase Order No. LO-726925. The effort was under the technical guidance of Mr. E.H. Johnson. It utilized Deep Space Network DSN/Mark 3 anomaly data compiled by Ms. Connie Johnsen. The efforts of the reviewers of this study are acknowledged as are those of Mrs. Ann Mellem and Mrs. Nancy Boschetti who prepared the manuscript. Our thanks also to SoHaR, Incorporated for their efforts in preparing the error taxonomy, analyzing the DSN anomaly data, summarizing the data and providing suggestions for further analysis.

DISCLAIMER

The findings of this report should not be misconstrued as representing an official Jet Propulsion Laboratory position. It is published only for the exchange and stimulation of ideas.

SECTION 1

INTRODUCTION

The purpose of this section is to introduce the reader to the goals, scope and importance of this study. In addition, this section defines terminology used within this report.

1.1 Problem Definition

Numerous studies have been conducted attempting to provide quantitative data on software errors (see References 1, 2, 3, 4). All of these studies collect and/or analyze error data taken from relatively large software development projects. These efforts are deemed important for the following reasons:

- A major item impacting costs, risks and uncertainty in software development is the lack of knowledge of what causes errors, why they occur and how they can be reduced (or at least located more quickly). The development of error data bases for software is a step towards the statistical quantification of error occurrence. Once error occurrences can be quantified, steps can be taken to reduce them.
- Identification of relationships between error occurrences, causes, criticality and time of error occurrence can lead to improved methods of detecting errors before they become difficult and costly to correct.
- Reliable error data can be used to measure the impact (both positive and negative) of modern software development and validation methodologies and tools on quality and productivity.
- The formal error documentation process forced by error data collection itself can provide better error control and help assure appropriate corrective actions are taken.

In the case of DSN/Mark 3, anomaly reports have been maintained for the last several years. However, since these reports were not intended for analysis, they were not based upon a consistent set of definitions for either the terminology or specific categorization types. In order to

learn from the DSN/Mark 3 effort and lower the costs of the forthcoming Network Consolidation Program (a huge program which will require over 426,000 lines of code), these software anomaly reports should be reduced to provide planners with a consistent software error data base which they can use to avoid critical anomalies which have occurred in the past.

1.2 Purpose

The purpose of this study was to develop a software error data base from software error records collected by NASA/JPL on a large command and control network called the Deep Space Network (DSN). This data base took existing DSN error information and categorized it so that errors within a class can be evaluated and classes of errors can be compared and analyzed. In addition, attempts were made to standardize the reporting. Finally, the data base will be used in subsequent studies to pinpoint those areas that most often cause software errors. Reduction in rework and improvement in quality should result when the data base information produced is used to build in safeguards against critical errors in future DSN development activities.

1.3 Scope

The results of this study are based upon our detailed analysis and interpretation of approximately 1000 DSN/Mark 3 anomaly reports. While these results are applicable to the DSN, they may not be generally applicable to the community at large. In addition, the trends identified by our cursory analysis have not been statistically evaluated. As a result, the tendencies revealed in our histograms may be deceptive. We, therefore, suggest that the reader wait until the results of the statistical analysis are published before he/she uses the data to justify error reduction projects.

1.4 Definitions

- Anomalies - irregularities, inconsistencies or other software imperfections.
- Deep Space Network (DSN/Mark 3) - a Jet Propulsion Laboratory ground based command and control system used to monitor and guide unmanned interplanetary space craft.

- Deep Space Network (DSN/Mark 4) - a network consolidation program that will combine the JPL DSN/Mark 3 network and the NASA Goddard STDN network.
- DSN/RCI Software Error Taxonomy - a methodology for categorizing Deep Space Network (DSN/Mark 3) errors.
- Error Category - an error cause defined by the DSN/RCI Software error taxonomy.
- Error Criticality - a error severity classification defined by the DSN/RCI software error taxonomy.
- Error Taxonomy - a set of rules for classificating errors.
- Error Time of Occurrence - a time when a software error was discovered classification defined by the DSN/RCI software error taxonomy.
- High Level Module Testing - testing of several modules together after low level module testing has been successfully performed.
- Module - a program unit that is discrete and identifiable with respect to combining with other units.
- Reliable Software - software that is correct (capable of executing and yielding correct results) and that meets other requirements such as timing and interfacing with the environment.
- Software Development - the formal process during which needs are transformed into a tested, documented, operational software system.
- Software Life Cycle - the period of time in which software is conceived, developed and used.
- Software Package - software delivered in product form to the potential user.
- Software Tool - computer software used to develop and maintain other computer software and/or documentation.
- Software Quality - the measure of the goodness of a software package in terms of meeting the objectives of the package.
- Testing - a process that examines the ability of an item to conform to the item's standards.

- Unit Testing - testing individual modules for correctness without allowing interactions with other modules.
- Verification/Validation - the process of determining the computer program was developed in accordance with stated specifications and satisfactorily performs the functions for which it was designed.

SECTION 2

MANAGEMENT SUMMARY

The purpose of this section is to provide an overview of the study; the six steps taken during its course (see Figure 2-1) and the study findings.

Over the life of the DSN/Mark 3 system, JPL has maintained records of anomalies as they occurred. These anomaly reports contain information about the error, including the phase in which it was discovered, its criticality, its disposition and a description of its effects. These reports were mainly used to assure that test and development personnel communicated and took appropriate corrective actions. The reports were maintained for several years in their raw form. Little analysis of error trends was performed. With the development of an even more complex system (the DSN/Mark 4 Digital Processing and Communications System) in the wings, the need for further analysis of these anomaly reports arose. This additional analysis could help the Jet Propulsion Laboratory define tools, checks and balances and procedures that have the potential to lower the expected error rate of future DSN/Mark 4 developments. Lower error rates could then result in lower software life cycle costs.

The bulk of this section summarizes the six steps taken in completing the first phase of this analysis (see Figure 2-1) which was aimed at building a consistent software error data base. Initial study findings are also discussed.

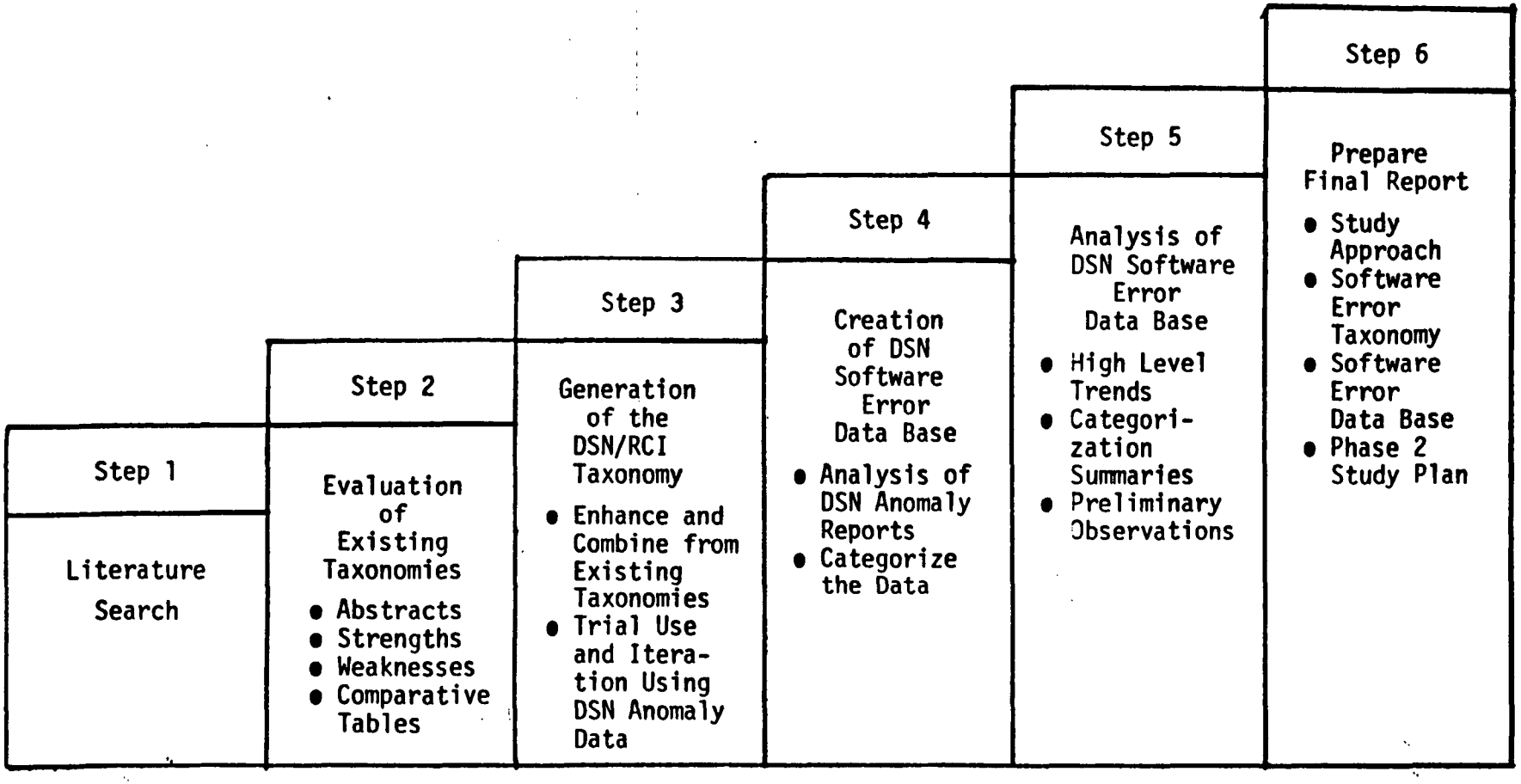
2.1 Summary of Study Approach

The six step study approach illustrated as Figure 2-1 was used during the course of the study. Each step will be described briefly in turn.

The aim of the first step was to identify previous work on software error taxonomies which we could capitalize on. An extensive search of literature was conducted and several candidate taxonomy developments were identified. An annotated bibliography (Reference 5) was then published so that the results of our effort could be captured for others to use.

Figure 2-1

Study Approach



The second step was conducted to evaluate the potential utility of the error classification schemes identified in step 1 for the DSN effort. Each taxonomy was studied and evaluated for strengths and weaknesses relative to the goal of capturing meaningful statistics in the areas of error category, error criticality and time of error occurrence. The result of the evaluation indicated that a new scheme had to be devised to meet the specific objectives of the DSN.

During the third step, a new DSN/RCI software error classification scheme was developed and validated by trial use. This new scheme (Reference 6) integrated together the best facets of the existing taxonomies in a manner acceptable to the DSN. The scheme allowed comparison with statistics generated by the previous efforts.

The fourth step created a DSN software error data base. Approximately 1000 DSN/Mark 3 anomaly reports were analyzed and then categorized in terms of their category, criticality and time of occurrence using the DSN/RCI software error taxonomy developed during step 3.

Once the data base was created, a preliminary analysis of it was conducted as step 5. Summaries of the data were compiled and evaluated so that recommendations for improvement could be formulated.

The final step was aimed at producing this report. As stated in the introduction, this document presents our findings and recommendations and culminates our study effort.

2.2 Overview of Study Findings

The major conclusions of this study can be summarized as follows:

- Software error data is an important management tool because it indicates where problems exist and where management attention should be placed.
- Existing software error taxonomies could be refined and adapted to capture meaningful error data from existing DSN error reports.
- Preliminary analysis of the DSN data base indicates that the majority of critical errors occurred under the requirements and design

categories. Studies into the use of independent verification and validation and better review and documentation procedures were recommended to contend with this phenomenon.

- Our cursory evaluation of the data also indicated that a large number of "other" errors were reported. These can be attributed to poorly defined man/machine interfaces, inadequate anomaly reporting procedures and no exception handling procedures being in place. Recommendations for studies into correction of these problem areas were also included in this report.
- Further analysis of the DSN error data base was also recommended so that insight into the cause and possible cures of errors could be determined based upon a detailed statistical analysis of the data collected during this effort.

SECTION 3

STUDY APPROACH

The purpose of this section is to provide detailed discussion of each of the six steps performed during the course of the study (see Figure 2-1).

3.1 Literature Search

We started our study by conducting an extensive search of existing literature. This search was performed for the following reasons:

- To locate existing taxonomies that were candidates for building the DSN software error data base.
- To provide a baseline for refinement if none of the above were directly applicable.
- To provide guideline targets for comparing DSN error data to software error data categorized using other classification schemes.

The search located several software error taxonomies. These were abstracted and evaluated in terms of their strengths and weaknesses in order to objectively compare them to the DSN needs. The output of this task was published and is included as Attachment A to this report. The major taxonomies identified by our literature search were:

- Logicon
- NASA/GSFC
- NASA/LaRC
- TRW

Six other works that impacted our development of an error taxonomy for DSN were also identified. Because these works were either near twins of the major taxonomies listed above or were applications of these taxonomies, they are not listed. They are, however, annotated in Attachment A of this report.

3.2 Evaluation of Existing Taxonomies

Our next task was to evaluate the existing software error taxonomies. This evaluation was performed for the following reasons:

- To evaluate whether or not the existing taxonomies could handle the DSN anomaly report data.
- To evaluate whether or not the taxonomy was simple to use and understand.
- To determine whether the classification system was based upon a consistent set of definitions for key terminology.
- To assess whether the taxonomy could take advantage of all the most useful information from the existing DSN anomaly reports. These data included time of error occurrence, error criticality, error category (through evaluation of the anomaly report text) and other information.
- To determine whether the taxonomy encapsulated information contained within other taxonomies. The ability to map to other taxonomies was considered important because it allows DSN error data to be compared with other existing error data. Such comparisons identify how well the DSN does in terms of other's experiences.

There was a great deal of commonality among the existing taxonomies. However, each had some form of uniqueness. The commonality can be clearly seen by referring to the following tables:

- Table 3-1, Taxonomy Handling of Time of Error Occurrence
- Table 3-2, Taxonomy Handling of Error Criticality
- Table 3-3, Taxonomy Handling of Error Category

Although each evaluated taxonomy had some strengths, it was decided that none of them were sufficient to meet the needs of the DSN without major rework. Therefore, the best attributes of each of the taxonomies were combined and a new taxonomy was developed. Tables 3-1, 3-2 and 3-3 illustrate how the taxonomies were integrated into a common classification framework.

The final evaluations of the major existing taxonomies are described in the following subparagraphs.

3.2.1 Logicon Error Taxonomy Evaluation

The major aim of the Logicon error study was to create a validation phase error data base (stressed correctness of already developed systems). They did not emphasize data collection or reduction during development which was felt to be a major weakness. In addition, the study tried to encom-

TABLE 3-1

TAXONOMY HANDLING OF TIME OF ERROR OCCURRENCE

TRW	LOGICON	NASA/LaRC	NASA/GSFC	DSN/RCI
• Requirements	• None	• None	• Requirements	• Development (Design or Programming Error)
• Design			• Functional Specs	• Verification (High Level Module Testing)
• Coding			• Design	• Acceptance (Formal System Test)
• Maintenance			• Coding & Test	• Transfer
• Unknown				

TABLE 3-2

TAXONOMY HANDLING OF ERROR CRITICALITY

TRW	LOGICON	NASA/LaRC	NASA/GSFC	DSN/RCI
• Critical	• None	• System Crash	• None	• Critical
• Serious		• Dependent Job Failure		• Dangerous
• Moderate		• Local Job Failure		• Minor
• Trivial		• Other		

TABLE 3-3

TAXONOMY HANDLING OF ERROR CATEGORY

	TRW	LOGICON	NASA/LaRC	NASA/GSFC	DSN/RCI TAXONOMY
1.	Computational	Computational	Computational	Control Logic or Computation	Computational
2.	Logic	Branch & Jump Logic & Sequencing	Logic	--	Logic
3.	Data Input	--	Data Input	--	(Under Data Handling)
4.	Data Handling	Data/Instruction/ Access & Storing	Data Handling	Misunderstanding of External Environment	Data Handling
5.	Data Output	--	Data Output	--	(Under Data Handling)
6.	Interface	Interrupt Ability & Data Coherency	Interface	--	Interface
7.	Data Definition	Incorrect Constant Value & Data Formats	--	--	(Under Data Handling)
8.	Data Base	--	Data Base	Mistaken Assumption as to Value or Structure	Data Base
9.	Operation	--	Operation	--	Operation
10.	Other (Includes Clerical)	Violation of Programming Practices	Other	Other	Other
11.	Documentation	Documentation	Documentation	--	(Under Other)

TABLE 3-3 (cont'd.)

	TRW	LOGICON	NASA/LaRC	NASA/GSFC	DSN/RCI TAXONOMY
12.	--	Incomplete or Erroneous Specification	--	Requirements Incorrect or Misinterpreted	Requirements Incorrect
13.	--	Specification Violation Due to Incorrect Implementation	--	--	(Under Other)
14.	--	Erroneous Use of System Hardware/ Software	--	--	(Under Operation)
15.	--	--	Keypunch	--	(Under Clerical)
16.	--	--	JCL	--	--
17.	--	--	Array Processing	--	--
18.	--	--	Program Execution	--	--
19.	--	--	--	Error in Use of Programming Language/Compiler	(Covered in Part Under Operation)
20.	--	--	--	Functional Specs	(Under Requirements Incorrect)

TABLE 3-3 (cont'd.)

	TRW	LOGICON	NASA/LaRC	NASA/GSFC	DSN/RCI TAXONOMY
21.	--	--	--	Design Error (Several Components)	Design
22.	--	--	--	Design or Implementation of Single Component	(Under Design)
23.	--	--	--	Error Related to Previous Change	(Under Appropriate Category)
24.	--	--	--	Clerical	Clerical

pass error data taken from commercial, scientific, real-time, application and system software. This generality of application made the Logicon taxonomy difficult to apply to the DSN case. The Logicon scheme handled the three classification dimensions needed by the DSN as follows:

- Time of Error Occurrence
  This dimension was not identified by the Logicon taxonomy.
- Error Criticality
  Four error severity categories were identified:
  - Catastrophic
  - Serious
  - Moderate
  - Trivial

  These categories could have been directly applied to classifying DSN anomaly reports.
- Error Category
  Twelve major error categories were defined (see Table 3-3). However, each category has several subcategories for finer error breakdown. The large number of error type classifications makes this taxonomy complex for use in classifying the DSN anomaly reports.

While the handling of error criticality was nicely done, the typing was not, in our opinion. We, therefore, decided we could not use this scheme as is for the DSN without major modifications.

3.2.2 NASA/GSFC (Goddard Space Flight Center) Error Taxonomy Evaluation

The NASA/GSFC taxonomy was appealing since it was simple (contained on a two-page form) and captured a lot of useful data. This taxonomy was specifically designed for Goddard application. The activities used to validate the program, detect the error and find its cause were defined in a matrix so the person filling out the form could quickly check off those actions taken. This matrix showed such things as programmer interaction in addition to normal test runs. Errors that were caused by previous changes were noted since this was often a significant percentage.

The three taxonomy dimensions were handled as follows by the Goddard form:

- Time of Error Occurrence
  The classifications (requirements, functional specifications, design, and coding and test) used did not provide a sufficient breakdown for the DSN. The reason is that the DSN anomaly report data covered the entire software life cycle while the NASA/GSFC scheme was only intended to show when the error originally entered the system.

- Error Criticality
  This dimension was not specifically identified by the NASA/GSFC scheme.

- Error Category
  Eight major error categories (see Table 3-3) were defined in their two-page form, with two major sub-classifications for design and implementation errors. These classifications were interesting since they were from a different perspective than those used by the other taxonomies. However, we needed more precise definitions of the terminology so that we could statistically analyze the data base at a later time.

Again, we felt we could not use the Goddard approach as is to fulfill the DSN's needs. We also felt that we should encapsulate the Goddard scheme in whatever scheme we adopted or developed so that the DSN and Goddard error histories could be compared and evaluated in the future.

3.2.3 NASA/LaRC (Langley Research Center) Error Taxonomy Evaluation

The NASA/LaRC taxonomy was appealing because it provided a one-page failure report analysis form that was designed both for data capture and for future data analysis.

The three taxonomy dimensions were handled as follows:

- Time of Error Occurrence
  This dimension was not specifically identified by the NASA/LaRC taxonomy.

- Error Criticality
  The following five error severity categories were identified:
  - System crash
  - Dependent job failure

- Local job failure only
- Real time failure
- Other

These categories were not fully applicable to the DSN anomaly report analysis since they contained only criticality indices and not breakdowns of job failure types.

- Error Category

  The twelve error type classifications (see Table 3-3) used were similar to the major classifications used in the TRW taxonomy. We felt this high level classification system was preferable to the many sub-classifications used by TRW. However, the classifications could be reduced even further for the DSN's purposes.

Again, we felt we could not use the work as is. The error typing approach strongly influenced the classification scheme we finally agreed upon though.

3.2.4 TRW Error Taxonomy Evaluation

The TRW taxonomy was quite useful because it was developed iteratively by first using existing error data to define error classifications and then by refining these classifications based upon trial use. The TRW scheme was used as the basis of many taxonomies identified during our literature search.

The three taxonomy dimensions were handled by TRW as follows:

- Time of Error Occurrence

  The development phase where the error originally occurred was tracked by the TRW taxonomy. This classification scheme does not adequately classify the DSN anomaly reports because the classifications do not cover the total DSN software life cycle.

- Error Criticality

  This dimension was not specifically identified by the TRW taxonomy. However, the study did suggest that error severity should be included in any scheme used to collect reliability data.

- Error Category
  The main error type classifications (see Table 3-3) lend themselves to the DSN anomaly data. However, the large number of sub-categories were thought to be too cumbersome to use effectively. Since it is difficult for a person to remember all the categories, we felt consistent classification using the scheme would be a problem.

Again, we did not feel we could use the TRW scheme without some modifications. The categories were well thought out and the results of classification were meaningful to our study. Therefore, we were influenced by their approach.

3.3 Generation of the DSN/RCI Software Error Classification Scheme

Although each of the software error taxonomies we reviewed was inadequate, each contributed to the final DSN/RCI taxonomy in some way. However, the NASA/LaRC taxonomy contributed the most. This taxonomy was extremely useful because it was specifically developed for NASA projects, it lends itself towards computer analysis and it combined many of the best points from the other classification approaches.

The existing DSN anomaly reporting system (Reference 7) played another important role in development of the DSN/RCI taxonomy. This system, developed specifically for the DSN, provided the input data for our data collection activity.

Based upon our evaluation of existing taxonomies and the DSN anomaly reports, a three dimensional classification scheme was devised to capture meaningful error data in a manner suitable for additional statistical and trend analysis. The new scheme capitalized on the experience of others and incorporated the best of what existed. The three dimensional software error taxonomy developed is discussed in detail in Section 4 of the report.

Each of the dimensions are summarized as follows:

- Time of Error Occurrence
  Defines in which of the four DSN phases of the software life cycle the error occurred. The four times are: development, verification, acceptance or transfer.

- Error Criticality
  Defines in which level of severity the error could be categorized. The three levels of severity are: critical, dangerous and minor.

- Error Category
  Categorizes the cause of the error. The ten error types are: computation, logic, data handling, interface, data base, operation, requirements incorrect, design, clerical and other.

It is important to note the DSN/RCI error taxonomy was developed only for building the DSN error data base and is not intended for other applications without further adaptation and modification.

3.3.1 Generation of DSN/RCI Error Classification Form

The error classification form illustrated as Figure 3-1 was designed to provide all desired error information for a particular anomaly report on a single sheet. All information encompassing the three dimensions of the taxonomy (time of error occurrence, error criticality and error category) was specified so the researchers filling out the forms had the taxonomy defined in non-ambiguous terms. A space for comments was provided since an explanation of the rationale for categorization was desired when it was not clear from the anomaly reports.

In addition to the three dimension information, other pertinent information was captured. This included the date the anomaly was closed (resolved or rejected) a subsystem identification and a notation regarding whether the anomaly was related to a previous change. Knowledge of how many new errors were caused by correcting others was considered important because it could suggest changes into how modifications and/or repairs were processed.

No._____

1. Subsystem_____ Program ID_____

2. Anomaly No._____ Date Closed_____

3. Activity: Comments

- Design_____ _____
- Verification_____ _____
- Acceptance_____ _____
- Transfer_____ _____

RELATED TO A PREVIOUS CHANGE_____

4. Criticality: Comments

- Type A_____ _____
- Type B_____ _____
- Type C_____ _____

5. Error Category Comments

- Computation_____ _____
- Logic_____ _____
- Data Handling_____ _____
- Interface_____ _____
- Data Base_____ _____
- Operation_____ _____
- Requirements Incorrect_____ _____
- Design_____ _____
- Clerical_____ _____
- Other_____ _____

6. Name of Person Completing Form_____
(date)

Figure 3-1

DSN Software Error Classification Form

This Page Intentionally Blank

SECTION 4

THE DSN/RCI SOFTWARE ERROR TAXONOMY

The purpose of this section is to explain the three dimensional error classification scheme developed to collect and analyze the DSN/Mark 3 error data. This scheme, which is illustrated as Figure 4-1, encompasses error time of occurrence, error criticality and error type. Each of these three dimensions is discussed in detail in the following paragraphs.

4.1 Time of Error Occurrence

Time of occurrence is the first dimension of the DSN/RCI error classification scheme. This dimension was deemed important because it identifies the point in the product life cycle where the error was discovered.

Four time classifiers were chosen because they were compatible with the DSN anomaly report data provided as input. The classifiers are as follows:

(D) Development - Anomalies in this category were reported during the design, coding and module unit testing activities. Most required design or programming revisions to be made. Errors in the category typically dealt with design problems between modules or with functional limitations of design. An example follows:

> "A system was required to provide human readable error messages on a log device. Unfortunately, the function was not specified in either the requirements and design specification. The error was discovered during a design review and an anomaly report was opened. Under such circumstances, we would state that the anomaly had occurred during development."

(V) Verification - Anomalies in this category were reported during integration and testing activities. Most were specification deviations that required the code to be revised. An example follows:

> "Module X expects a true or false condition as input from module Y. Unfortunately, module Y has not been specified to provide the true or false input. A test identified this problem during testing and an anomaly report was written scoping the rework. Under such circumstances we would state that the anomaly had occurred during verification."

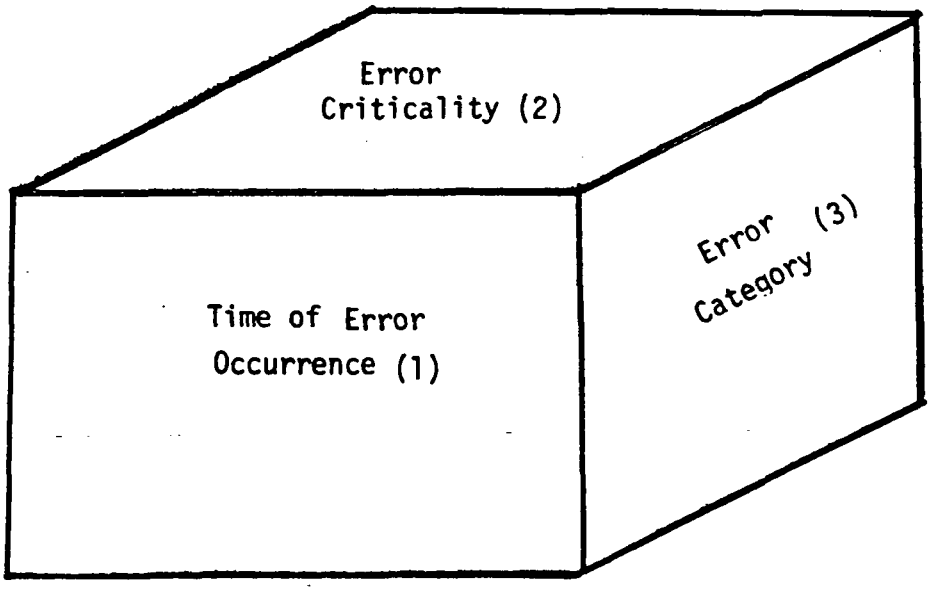


Figure 4-1

DSN/RCI Three Dimensional
Software Error Classification Taxonomy

(A) Acceptance - Anomalies in this category were reported during formal testing of the software. Errors in this category usually stem from requirements problems or improper mechanization. An example follows:

> "The system malfunctions when accepting more than six simultaneous inputs. The error was discovered during formal testing when the program was stressed and an anomaly report was written. Under such circumstances, we would state that the anomaly had occurred during acceptance."

(T) Transfer - Anomalies in this category were reported after the software package was put into operation in a live environment. These anomalies usually resulted from halts, failures or malfunctions. An example of such an anomaly follows:

> "The software halts when a zero input value is received. This error was discovered during operation when the DSN was reducing telemetry data. Under such circumstances, we would state that the anomaly occurred during transfer."

4.2 Error Criticality

Error criticality is the second dimension of the DSN/RCI error taxonomy. The three severity levels currently identified by the DSN anomaly report forms were retained to simplify the determination problem. The temptation to add a fourth category for trivial errors was resisted because trivial errors were often not recorded on the anomaly reports.

The three error criticality classifiers used are defined as follows:

- Level A - Critical error (error impacts mission performance or seriously degrades capability and no workaround exists). An example follows:

  > "The system halts when the value of one of its inputs exceeds its nominal end of range. Manual intervention is required before operation can be resumed. Under such circumstances, we would state that a level A error had occurred."

- Level B - Dangerous situation (error exists that could degrade performance or capability but a workaround exists). An example follows:

  > "A particular utility function causes the system to halt to await operator's action. The utility function is not required for correct system operation and can be disabled temporarily to correct the problem. Under such circumstances, we would state that a level B error had occurred."

- Level C - Minor problem (error exists that doesn't impact performance or capabilities and can be fixed at a more leisurely pace. An example follows:

  > "An informational message is displayed twice (rather than once) each time it is enabled. No other negative effect happens. Under such circumstances, we would state that a level C error had occurred."

4.3 Error Category

The third dimension of the DSN/RCI error taxonomy is error category. Each of the ten error categories was defined so that insight into the error causes could be ascertained. The ten categories are defined as follows:

1. Computation - Computation anomalies are errors in or resulting from coded equations. Examples of computation errors include: (a) Incorrect operand in equation, (b) Incorrect use of parenthesis, (c) Incorrect equation, (d) Missing computations and (e) Rounding or truncation error.
2. Logic - Logic anomalies are errors in sequencing, control or loop conditions. Examples of logic errors include: (a) Logic out of sequence, (b) Wrong variable being checked, (c) Missing logic or condition tests, (d) Too many/few statements in loop and (e) Loop iterated incorrect number of times.
3. Data Handling - Data handling anomalies are errors in handling input/output. Examples of data handling errors include: (a) Data initialization incorrect, (b) Variables not set properly, (c) Variable type incorrect, (d) Data packing/unpacking incorrect and (e) Subscripting error.
4. Interface - Interface anomalies are errors in communications between a routine and other routines, the data base and/or the user. Examples of interface errors include: (a) Data incorrectly transmitted from one routine to another, (b) Data incorrectly set/used from the data base, (c) Improper input/output synchronization and (d) Data sent to wrong destination.
5. Data Base - Data base anomalies are errors in preset data. Examples of data base errors include: (a) Data should have been initialized in data base but wasn't, (b) Data initialized to incorrect value and (c) Data base units are incorrect.

6. Operation - An operation anomaly is an error occurring as the software executes. Examples of operation errors include: (a) Operating systems errors, (b) Hardware errors, (c) Operator errors, (d) Compiler or support software errors and (e) Test execution errors.
7. Requirements Incorrect - Requirements errors deal with improper or ambiguous functional and software requirements specifications and not with implementation and/or operation. Software may correctly solve the wrong problem if it is specified improperly.
8. Design - Design errors deal with improper architectural and detailed design specifications which form the basis to which the program and the data base are mechanized.
9. Clerical - Clerical anomalies occur when people are involved in the translation. Examples of clerical errors include keypunch, typos and/or transliteration.
10. Other - Other is a "catch-all" for other types of error not encompasses by the scheme. Examples of other errors include incorrectly reporting that an anomaly had occurred when in reality it was a programmer misconception.

This Page Intentionally Blank

SECTION 5

STUDY FINDINGS

The purpose of this section is to discuss the major findings of this study. Each will be explained in the paragraphs that follow.

5.1 Utilization of Software Error Data

The collection and classification of software error data provide management with an important tool for spotting areas where problems exist and where management attention should be placed. This usefulness has been validated by several other error studies in addition to this study. For future DSN/Mark 4 programs, the classification of error data should be performed as the anomalies are reported. This would help assure that the error was more fully understood as it was reported. It would force personnel reporting errors to think about its causes, effects and criticality. The data itself could serve a useful purpose. First, it could be used to identify error-prone modules. Management could then concentrate attention and test resources to ensure problems don't develop when these modules are placed into operation. Second, the data could be used to serve as justification for making repair or replacement decisions. For example, a module would be replaced if a certain error threshold were exceeded. As another example, modules would not be accepted by the software librarian if an error threshold were exceeded by a programmer trying to get clean compiles with his/her program. Last, management could use the data to guide them during their test activities. For example, a program would not be placed into operation unless an acceptable error rate were being experienced. As another example, quality assurance would seed the program with errors and determine whether or not the test cases identified the majority of these. If the tests didn't, rework of the test program would be recommended.

We believe the DSN could make better use of the data it collects. A number of recommendations aimed at correcting current DSN practices are included within this report in Section 7. All are aimed at enabling DSN personnel to use the error data it collects as a management tool.

5.2 Control of the Anomaly Reporting Process

Our initial analysis of the DSN error data base identified a lack of uniformity and wide variance in how individual anomalies were classified. For

example, there are several cases where the same error was reported by different people who each classified the error differently. We believe this discrepancy can be corrected by modifying existing error reporting procedures currently in use, training DSN to better understand and use the procedures and making someone responsible for monitoring adherence to the procedures. The monitoring role is an important one. Feedback on procedures is necessary so they can be revamped and fine-tuned as necessary. Assurance is also necessary so that duplication can be minimized and forms can be properly filled out. Again, recommendations aimed at improving the error reporting process are included in Section 7 of this report.

5.3 Critical Error Occurrences

Analysis of the DSN software error data base indicates that many of the critical errors occurred during the requirements definition and design phases. These errors are the most costly to correct, especially if they are not caught early in the development cycle. Our preliminary analysis shows that up to seventy-five percent (75%) of the requirements errors are avoidable through better documentation techniques and that up to fifty percent (50%) of design errors may be caught by implementing better review techniques. Studies into the use of Independent Verification Validation (IV&V), better review and documentation procedures and automated tools are recommended in Section 7 of this report to contend with these problems.

It is important to realize that there is more than a subtle difference in roles for the IV&V and quality assurance organizations. Quality assurance is primarily a monitoring activity conducted to ensure standards are followed and specifications are met. IV&V is primarily an assessment activity conducted to ensure the integrity of the products as they are incrementally developed.

5.4 Nondescript Error Reporting

Our initial evaluation of the error data base showed the largest single error type classification was "other". Although some of these can be attributed to improperly completed anomaly reports, many of them could also be the result of poorly defined man/machine interfaces (e.g., commands that are difficult to use or whose incorrect usage causes the system to halt), improper and imprecise procedures for handling exceptions (anomaly reporting), inadequate documentation and/or user misconceptions (requests for enhancements/modifications that were not really anomalies at all).

Recommendations for studies investigating improving the DSN man/machine interface, developing a new DSN anomaly reporting procedure and implementing a DSN discussion/review form are included in Section 7 of this report. These studies should scope the magnitude of the problem and provide workable solutions.

5.5 Detailed Data Base Analysis

Our cursory analysis of the error data base has yielded some useful information about high level trends and problem areas. We believe this analysis to be incomplete and deceptive because it has not been statistically evaluated and compared with similar data taken from the public domain. We believe further analysis is necessary before information gleaned from the error data base can be used with confidence. Recommendations for further statistical analysis are therefore included within Section 7 of this report.

This Page Intentionally Blank

SECTION 6

ERROR DATA BASE

The purpose of this section is to summarize the DSN error data in terms of the DSN/RCI classification taxonomy. Histograms summarizing the 1000 anomaly reports are used for this purpose. In addition, the approach used to translate the anomaly report data into software error information is included for completeness.

6.1 Creating the Error Data Base

Each of the approximately 1000 DSN anomaly reports was carefully analyzed in terms of the DSN/RCI taxonomy's three dimensions. In cases where the categorization was not readily apparent, comments were included on the error classification form so that group concensus could yield a classification decision. Sometimes the information on the anomaly report was incomplete or unclear. Examples of such cases were where the problem and its resolution were described as symptoms (e.g., 'TIWIWT zeroed'). When this occurred, the person who originally completed the anomaly report was contacted and asked to clarify the rationale. Sometimes this procedure proved fruitless because persons contacted could not remember the circumstances of an error that occurred years ago. These problems were then resolved using the best judgment of the team.

It is important to note that the same people were involved in all classification decisions. As a result, consistency in interpretation was assured and any biases were factored uniformly across the sample size.

6.2 Data Summaries

Data itself is useless unless it can be reduced to yield meaningful information. RCI researchers took the error data base gathered by the study and generated histograms to identify apparent trends and conclusions without resorting to a detailed statistical analysis. The histograms combine error data within an accuracy range of plus or minus one percent. Seventeen histograms follow in subsequent paragraphs along with a discussion of our observations. To simplify the graphs the common abreviations listed in Table 6-1 were used consistently throughout this section.

TABLE 6-1

ABBREVIATIONS/ACRONYMS

- Time of Error Occurrence

  D - Development - design, coding and unit test of program modules

  V - Verification - integration and testing of subsystem

  A - Acceptance - formal testing and acceptance of subsystem

  T - Transfered - software subsystem operational

  U - Unknown

- Error Criticality Levels

  A - Critical

  B - Dangerous (work-around exists)

  C - Minor

  U - Unknown

- Errors By Error Category

  CO - Computational Error

  LO - Logic Error

  DH - Data Handling Error

  IN - Interface Error

  DB - Data Base Error

  OP - Operation Error

  RI - Requirements Incorrect

  DE - Design Error

  CL - Clerical Error

  OT - Other Error

TABLE 6-1 (cont'd.)

SUBSYSTEM NAME ABBREVIATIONS

DSS - Deep Space Systems

ODA - Occultation Data Assembly

NTK - Network Tracking RTM

CMF - Communication Monitor and Formatter

NTM - Network Telemetry

MDA - Metric Data Assembly

DRP - Data Records Processor

DST - Host Subsystem

DIS - Monitor Subsystem

NCD - Network Command

NDS - Digital Display Subsystem

NLP - Network Log Processor

NMC - Network Monitor and Control

NRS - Network Radio Science

VAP - Video Assembly Processor

VLB - VLBI Block I Program

APS - Antenna Pointing System

BRP - Breakpoint Processor

CPA - Command Processor Assembly

NOC - Network Operations Center

6.2.1 Errors by Time of Occurence

A histogram illustrating errors by time of occurence, Figure 6-1, was produced by counting the number of errors in the data base for each of the following activities: software development, software verification, software acceptance testing, software transfered to operational usage, and undefined time of occurrence. The undefined time of occurrences resulted from anomaly reports which had no time of occurrence specified, and for which time of occurrence could not be ascertained by our researchers. The preliminary observations we can make based on this histogram are as follows:

- The data seems to indicate that formal anomaly reporting procedures were not strictly enforced during the development of most of the subsystems investigated by this study.
- The software verification and acceptance testing processes uncovered a large number of errors. Unfortunately, there were still many more errors not discovered until the subsystem was placed into operation. Life cycle costs could have been lowered considerably by investing in methods and tools that located and isolated anomalies earlier in the process.

6.2.2 Errors by Criticality Level

A histogram illustrating errors by criticality level, Figure 6-2, was produced by counting the errors in the data base for each of the following three error criticality levels: A or critical level, B or dangerous level (can be worked around), and C or minor level. An additional classification U was included to identify those anomolies for which the error criticality could not be ascertained by our researchers. The preliminary observations we can make based on this histogram are as follows:

- Level B errors were in the majority. Although work arounds could be devised, such a large number of errors makes existing quality assurance practices suspect.
- A large number of level A errors were identified. Critical errors of such a large proportion immediately call attention to review procedures and testing approaches used during development. Some review of these might be in order.

FIGURE 6-1

ERRORS BY TIME OF OCCURENCE

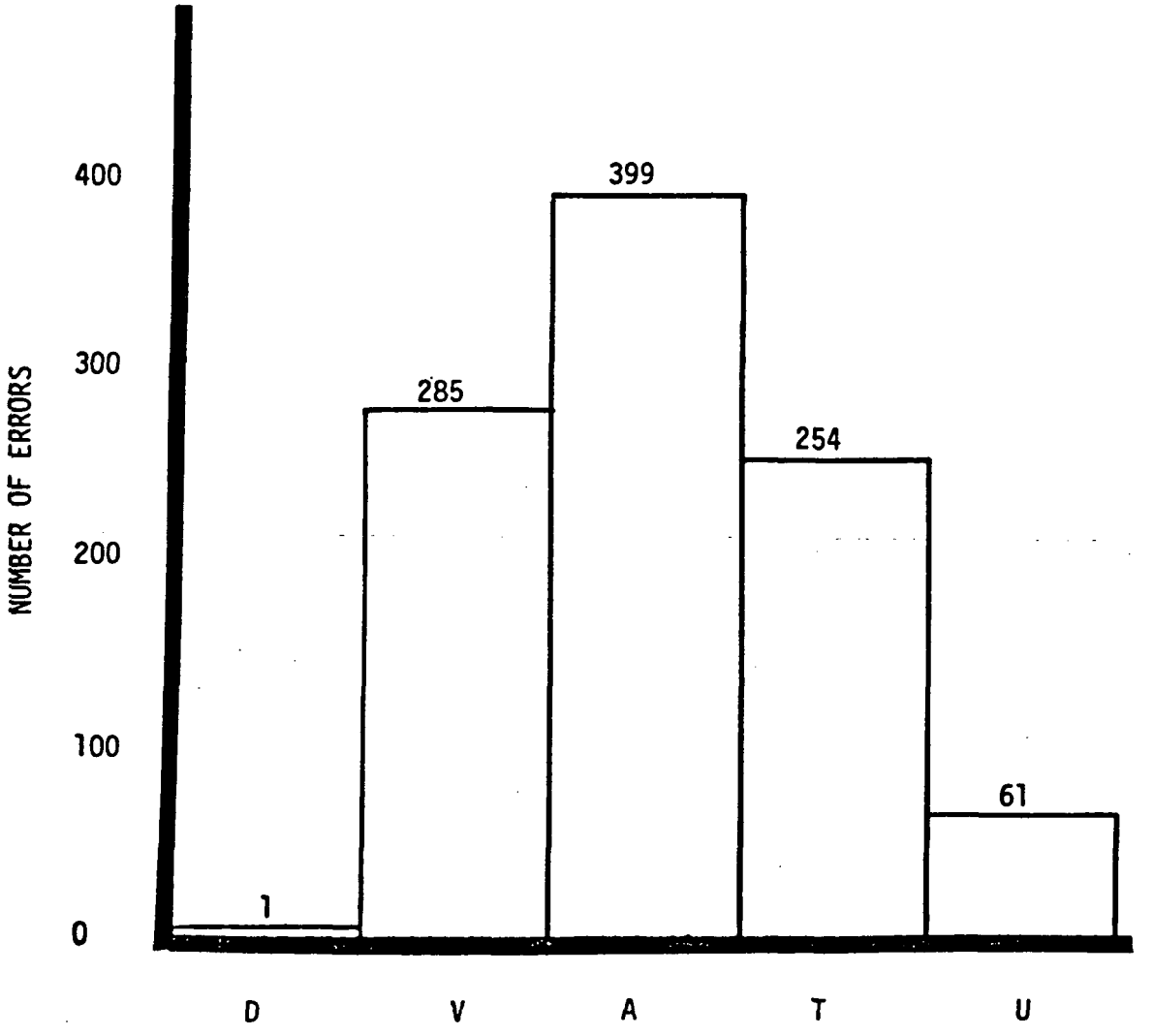
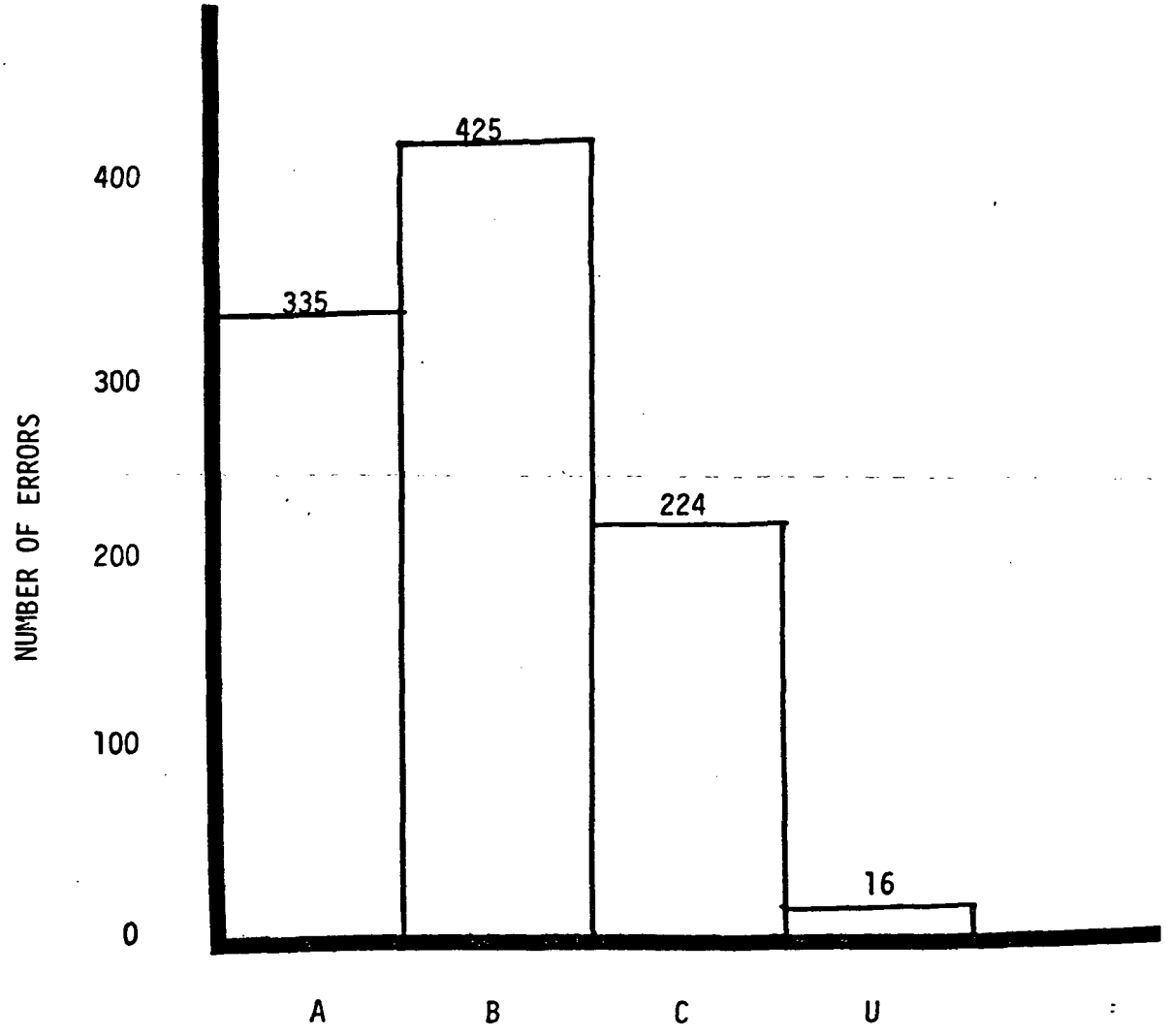


FIGURE 6-2

ERRORS BY CRITICALITY



6.2.3 Errors by Category

A histogram illustrating errors by error category, Figure 6-3, was produced by counting the errors in the data base for each of the following ten RCI/DSN error categories: computation, logic, data handling, interface, data base, operational, requirements incorrect and design clerical, and other. An additional classification "questionable" consists of "other" anomalies for which no change was generated. These "questionable" errors were the subset of "other" errors that resulted from documentation requests, grips, misunderstandings, politics, and potential hardware failures. The preliminary observations we can make based on this histogram are as follows:

- Design and requirements errors were the largest single source of anomalies (other is smaller if the questionable anomalies are eliminated). Many of these errors could be eliminated in the future by using tools for design and requirement analysis and an independent verification and validation contractor.
- Some anomalies of the "questionable" subcategory of "other" were not errors but really requests for changes or documentation. This seems to indicate the need to improve existing anomaly reporting procedures and the mechanisms used for quality control. The data also indicates the need for reviewing each anomaly report to suppress phantoms before they are processed.

6.2.4 Level A Criticality by Subsystem

A histogram illustrating level A criticality by subsystem, Figure 6-4, was produced by counting all the level A (critical) errors in the data base for each subsystem for which there was reported data. Some subsystems with no level A errors reported were not plotted. The preliminary observations we can make based on this histogram are as follows:

- Subsystems DSS, CMF, and CPA produced the largest numbers of critical errors. The functions performed by these subsystems are probably the most complex and should be more carefully reviewed and more strenuously tested during future developments.
- Because level A errors are critical, special procedures should be established to review subsystems that experience extremely high

FIGURE 6-3

ERRORS BY CATEGORY

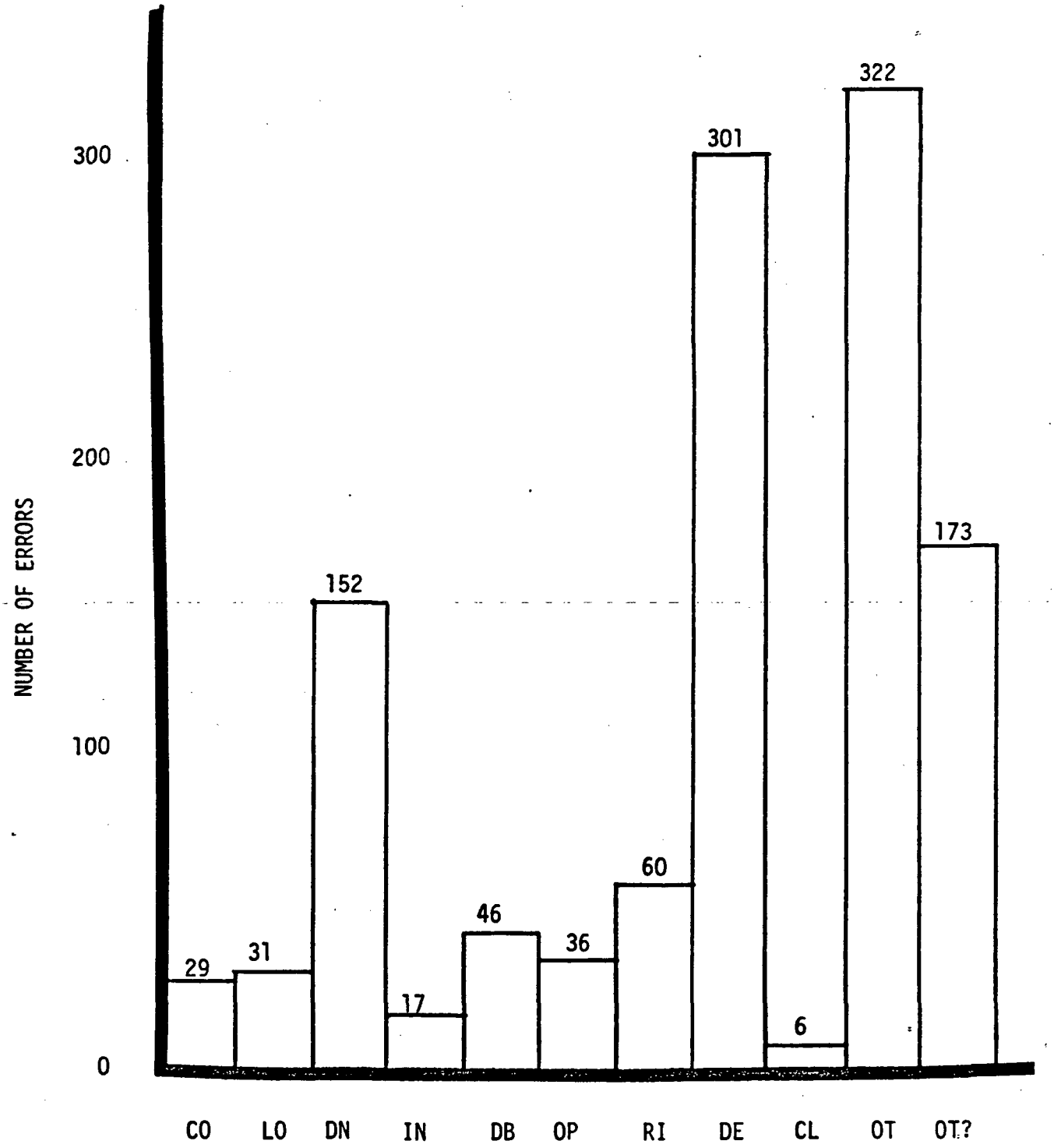
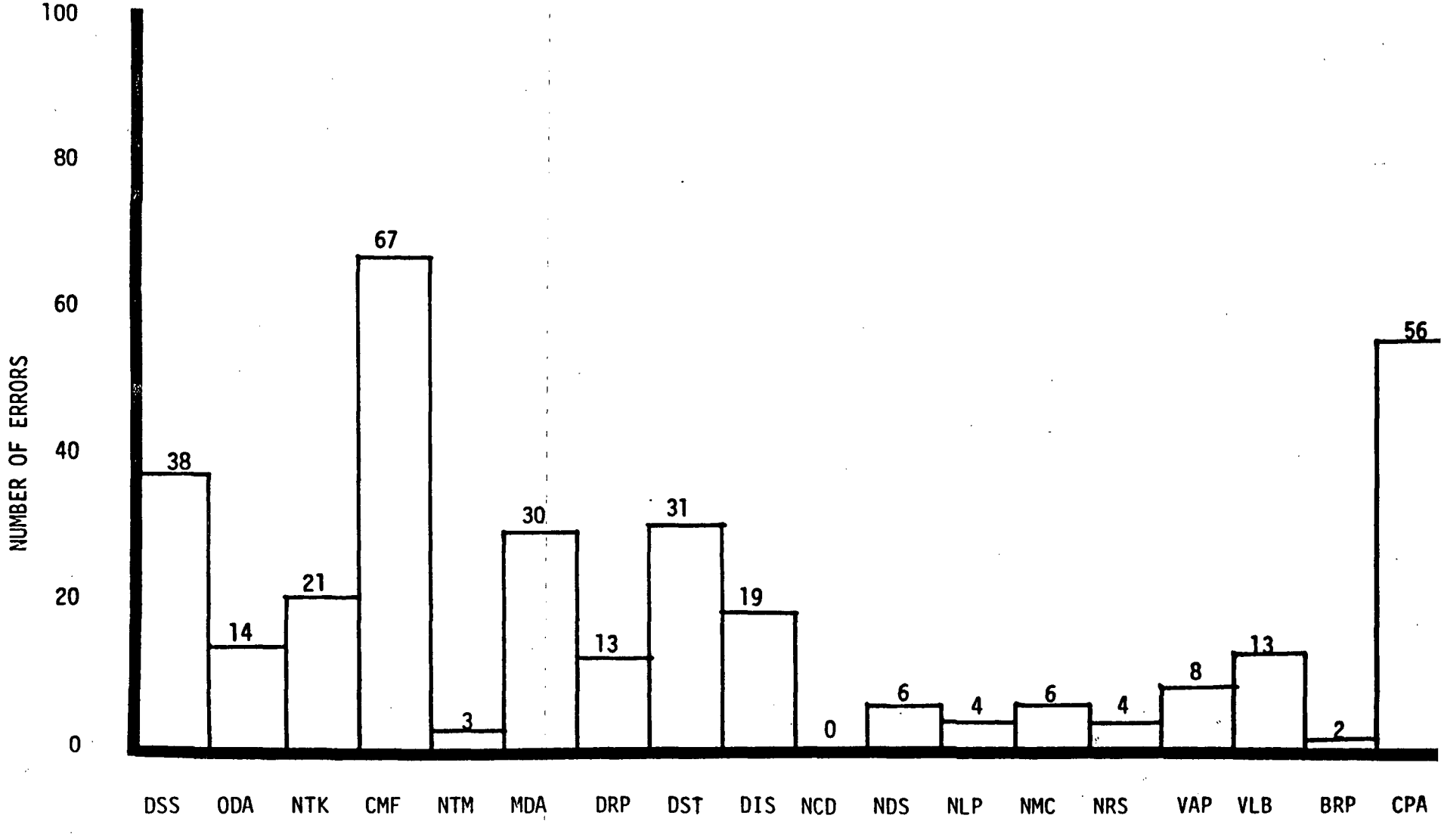


FIGURE 6-4

LEVEL A CRITICALITY BY SUBSYSTEM



incidences of errors. For example, error-prone analysis might be warranted to understand in-depth why a certain threshold of errors were exceeded during a specific time period.

6.2.5 Level B Criticality by Subsystem

A histogram illustrating level B criticality by subsystem, Figure 6-5, was produced by counting all the level B (dangerous) errors in the data base for each subsystem for which there was reported data. Some subsystems with no level B errors reported were not plotted. The preliminary observations we can make based on this histogram are as follows:

- Subsystems DSS, NTK, CMF, and CPA produced the largest numbers of dangerous errors. These subsystems were also the same ones that had the highest incidences of level A errors. This data emphasizes the need to have these subsystems analyzed in-depth to find out why they are so error-prone.
- Further analysis of these trends is warranted and provided in subsequent subparagraphs.

6.2.6 Level C Criticality by Subsystem

A histogram illustrating level C criticality by subsystem, Figure 6-6, was produced by counting all the level C (minor) errors for each subsystem in the data base. The preliminary observation we can make based on this histogram is as follows:

- Most errors were identified as level A or B leaving this category immediately suspect. Because this category is the lowest priority, the tendency was to elevate anomalies to more critical classifications. This probably resulted in a large number of errors being misclassified. Again, better review procedures for anomalies can help rectify this situation. Also, it probably led to a large number of minor errors not being reported.

6.2.7 Errors During Verification by Error Category

This histogram, Figure 6-7, was produced by counting all the errors for each of the ten error categories that occured during the verification (software integration and testing) process. It shows which of the error

FIGURE 6-5

LEVEL B CRITICALITY BY SUBSYSTEM

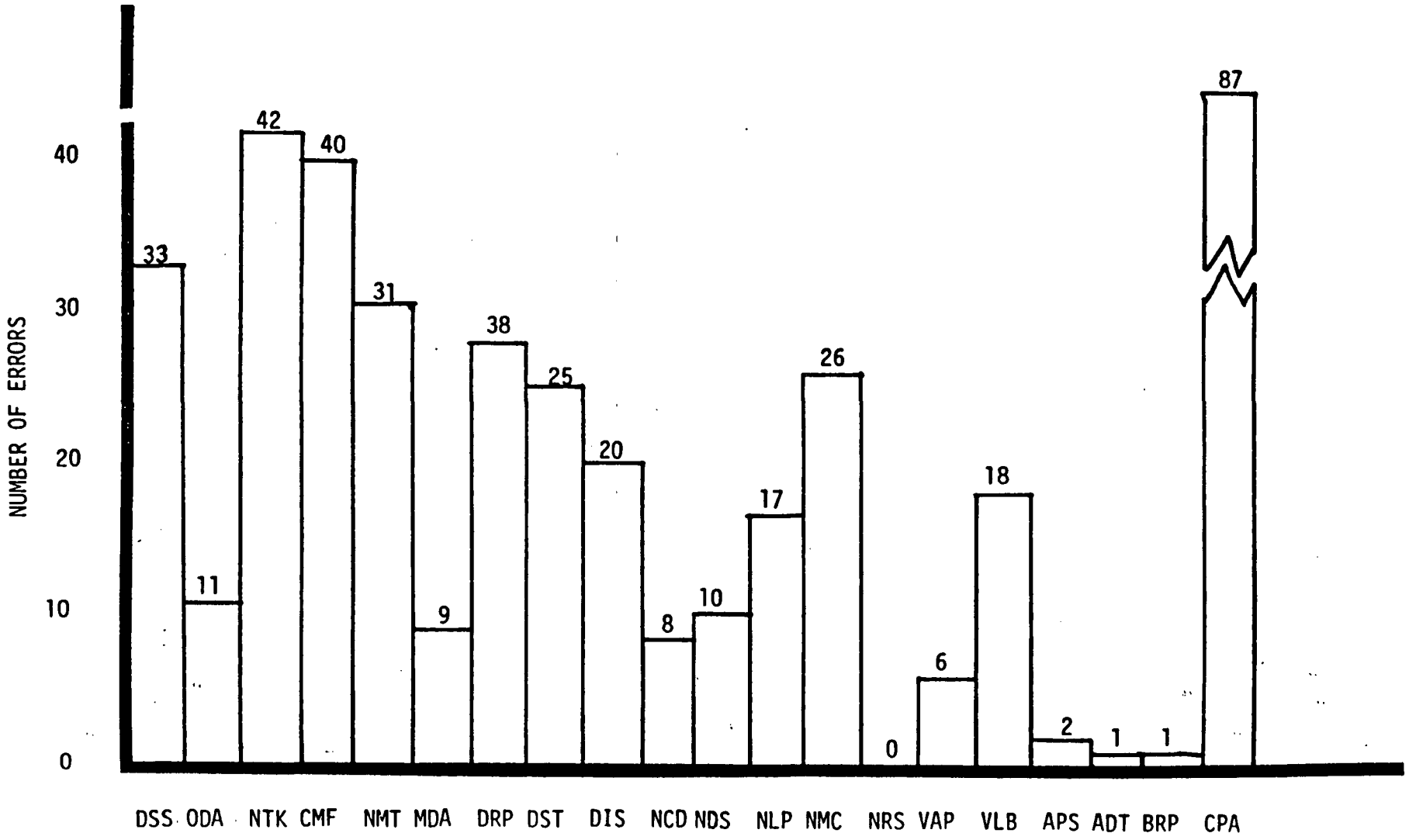


FIGURE 6-6

LEVEL C CRITICALITY BY SUBSYSTEM

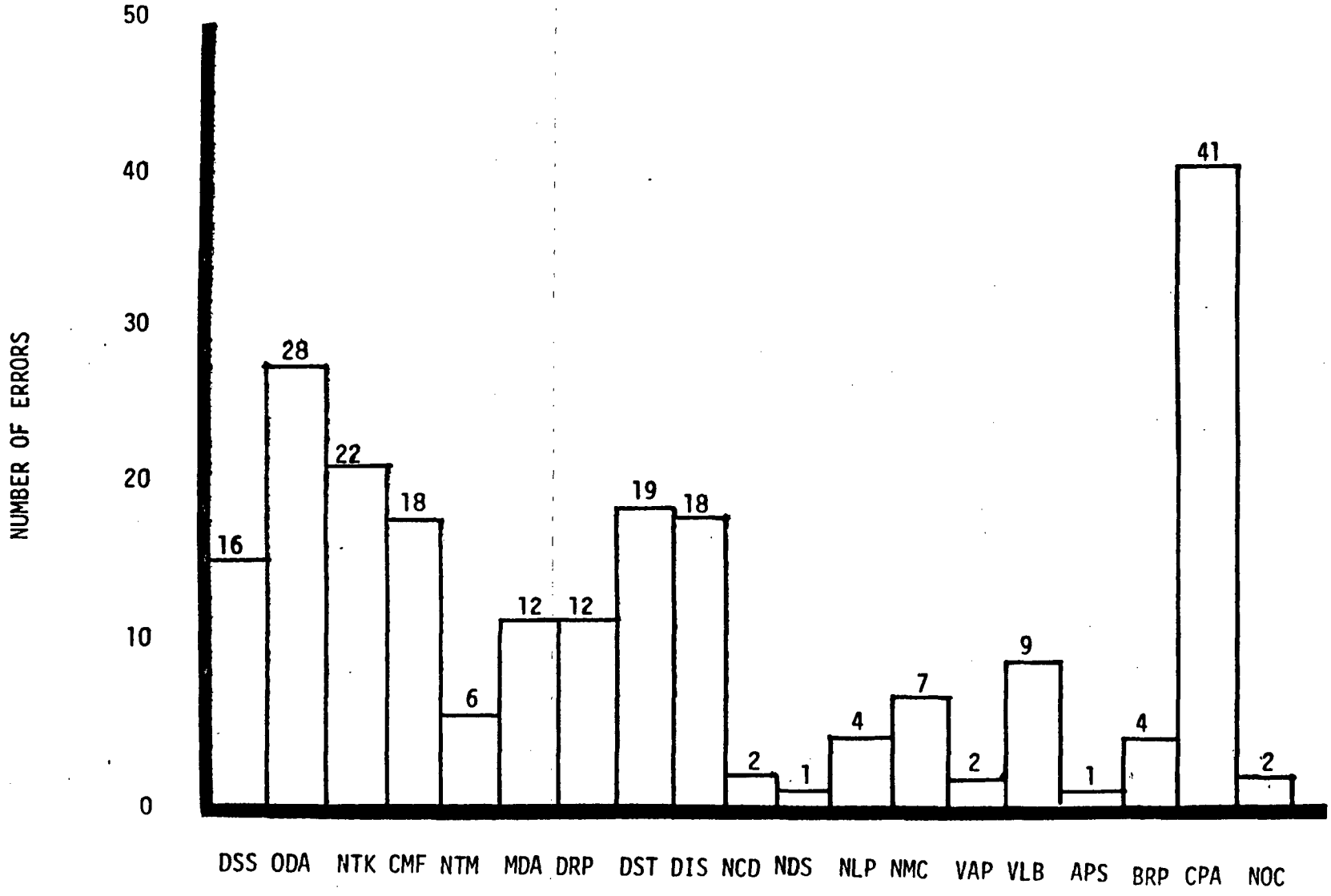
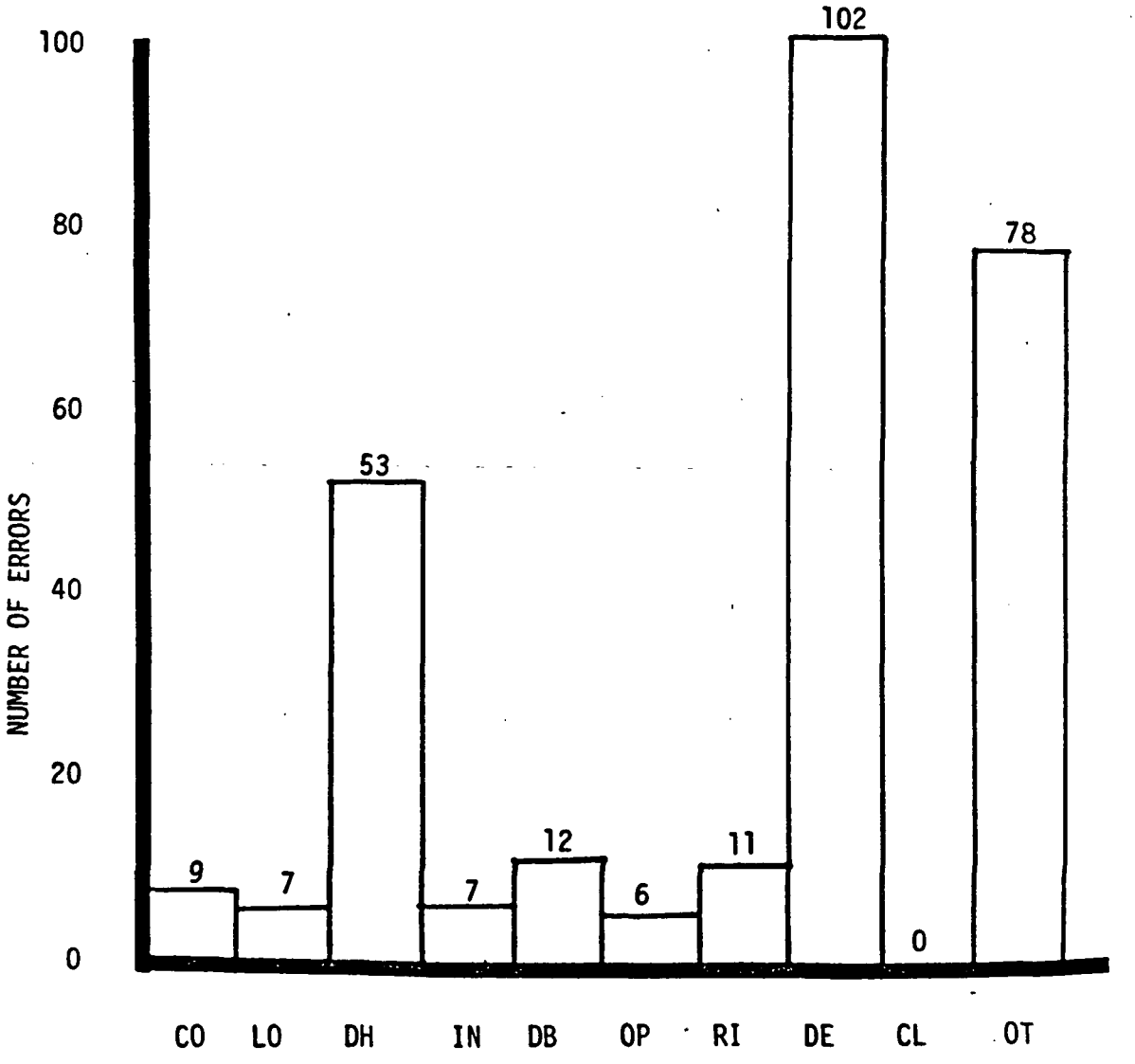


FIGURE 6-7

ERRORS DURING VERIFICATION - BY ERROR CATEGORY



categories caused errors while the software was subjected to high level integration and testing. The preliminary observations we can make based on this histogram are as follows:

- Design errors were the most prevelant classification during software integration and testing. Many of these design errors could be eliminated earlier in the life cycle and at reduced cost using automated design analysis tools. An independent verification and validation contractor or stronger quality assurance involvement earlier in the project could probably help as well.
- Data handling errors were common during software integration and testing. Many of these could be eliminated by educating programmers on data interfacing standards and by automatically checking compliance of all programs to data handling conventions during unit check-out. The Air Force uses such an approach at their Satellite Test Center in Sunnyvale, California to reduce such error incidences.
- Again, the large percentage of "other" errors could probably be reduced by procedural improvements.

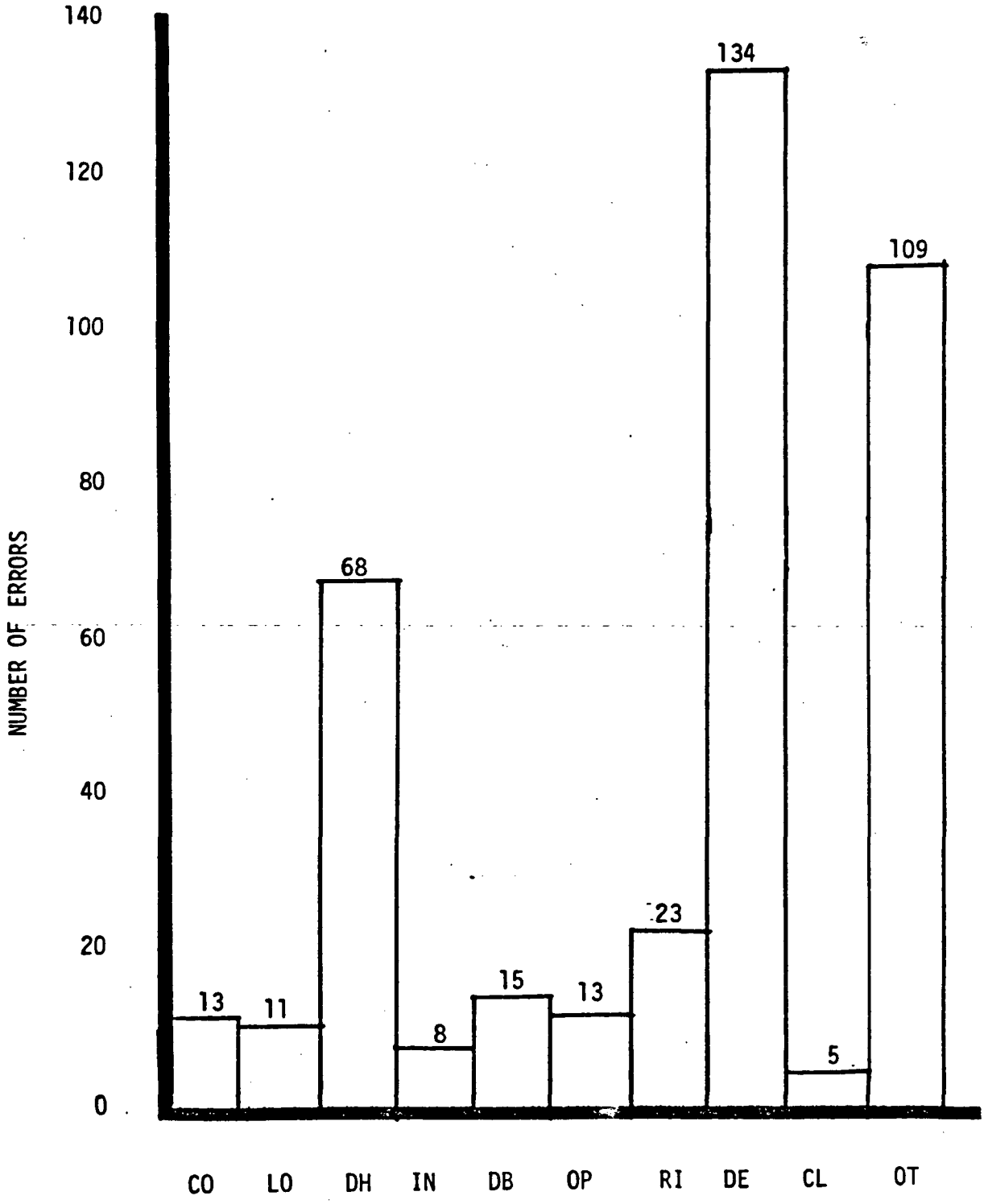
6.2.8 Errors During Acceptance Testing by Error Category

This histogram, Figure 6-8, was produced by counting all the errors for each of the ten error categories during the acceptance (formal software testing before transfer to the user) process. The preliminary observations we can make based on this histogram are as follows:

- Design errors are again the most prevelant classification during formal acceptance testing. Many of these design errors should have been eliminated well before this activity commenced. This data stresses the need for investigating earlier forms of design analysis to eliminate errors before they propagate in impact and cost.
- Data handling errors are again common during formal software acceptance testing. Our suggestion to improve training and automatically check for compliance during unit testing seems to make sense.

FIGURE 6-8

ERRORS DURING ACCEPTANCE - BY ERROR CATEGORY



6.2.9 Errors After Transfer by Error Category

This histogram, Figure 6-9, was produced by counting all the errors for each of the ten error categories after the software was transfered to the user (made operational in a production environment). It shows which of the error categories caused errors after the software was declared operational. The preliminary observations we can make based on this histogram are as follows:

- "Other" errors are the most prevalent classification. These errors included requests for modification, documentation and clarifications. The high percentage of these errors indicate that the user does not seem to fully understand what the delivered system is supposed to do and how he/she should interface with it. This human interface problem (i.e., man-machine interface) is critical to proper operation and seems to deserve immediate attention. Also, conventions for user documentation should be investigated to determine whether or not improvements are in order.
- Again, there were a large number of design errors. Our previous observations and recommendations should reduce this large number of incidences.

6.2.10 Level A Criticality by Error Category

This histogram, Figure 6-10, was produced by counting all the level A (critical) errors for each of the ten error categories. The histogram identifies which of the error categories caused which proportion of level A error incidences. The preliminary observations we can make based on this histogram are as follows:

- Design errors seemed to cause a large number of level A or critical errors. This provides us with further evidence of the need to investigate earlier detection of design errors.
- Data handling errors were also a cause of a large number of level A errors.
- Surprisingly, "other" errors contributed a large number of level A errors. This could be attributed to the user who could not operate

FIGURE 6-9

ERRORS AFTER TRANSFER - BY ERROR CATEGORY

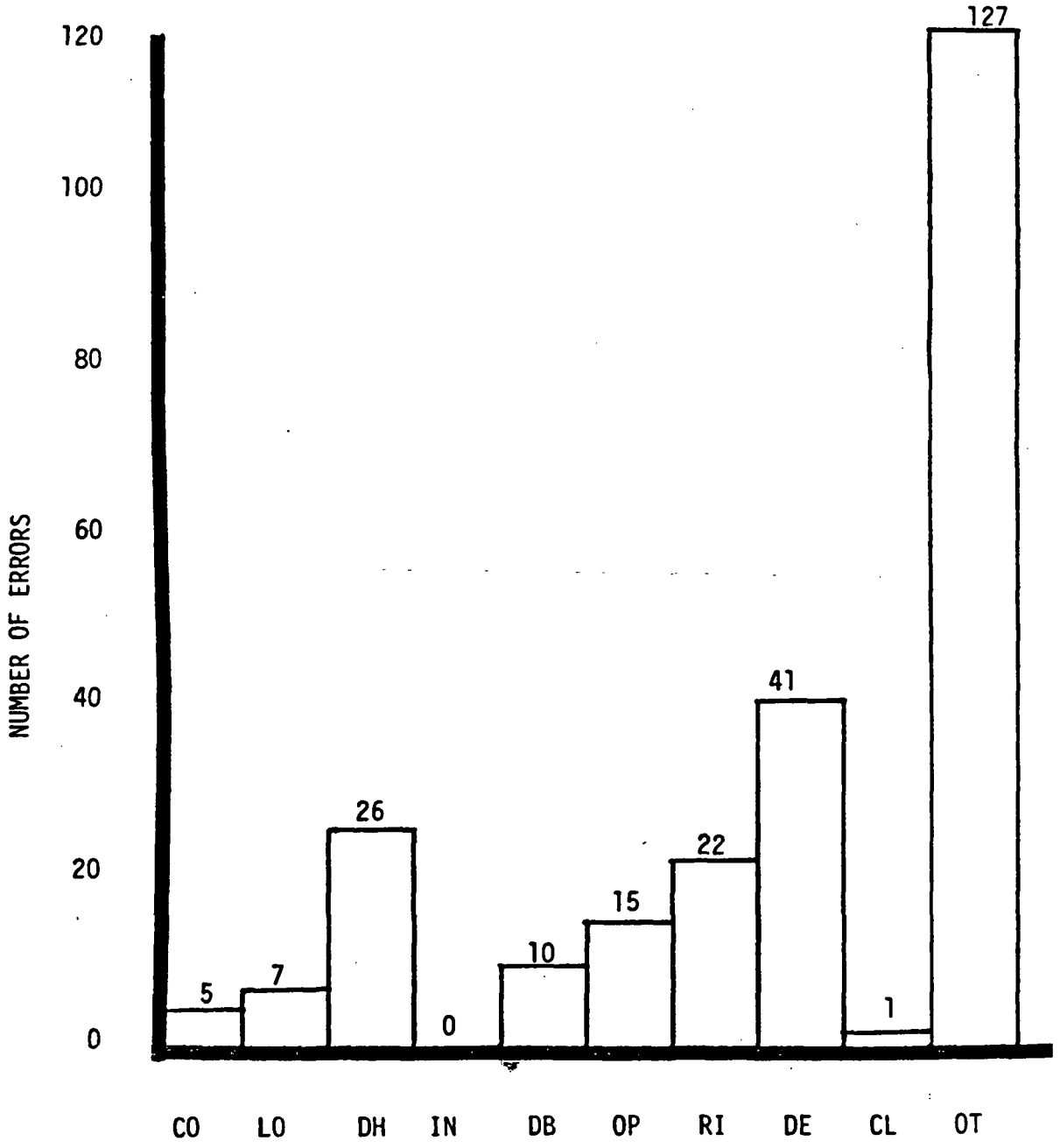
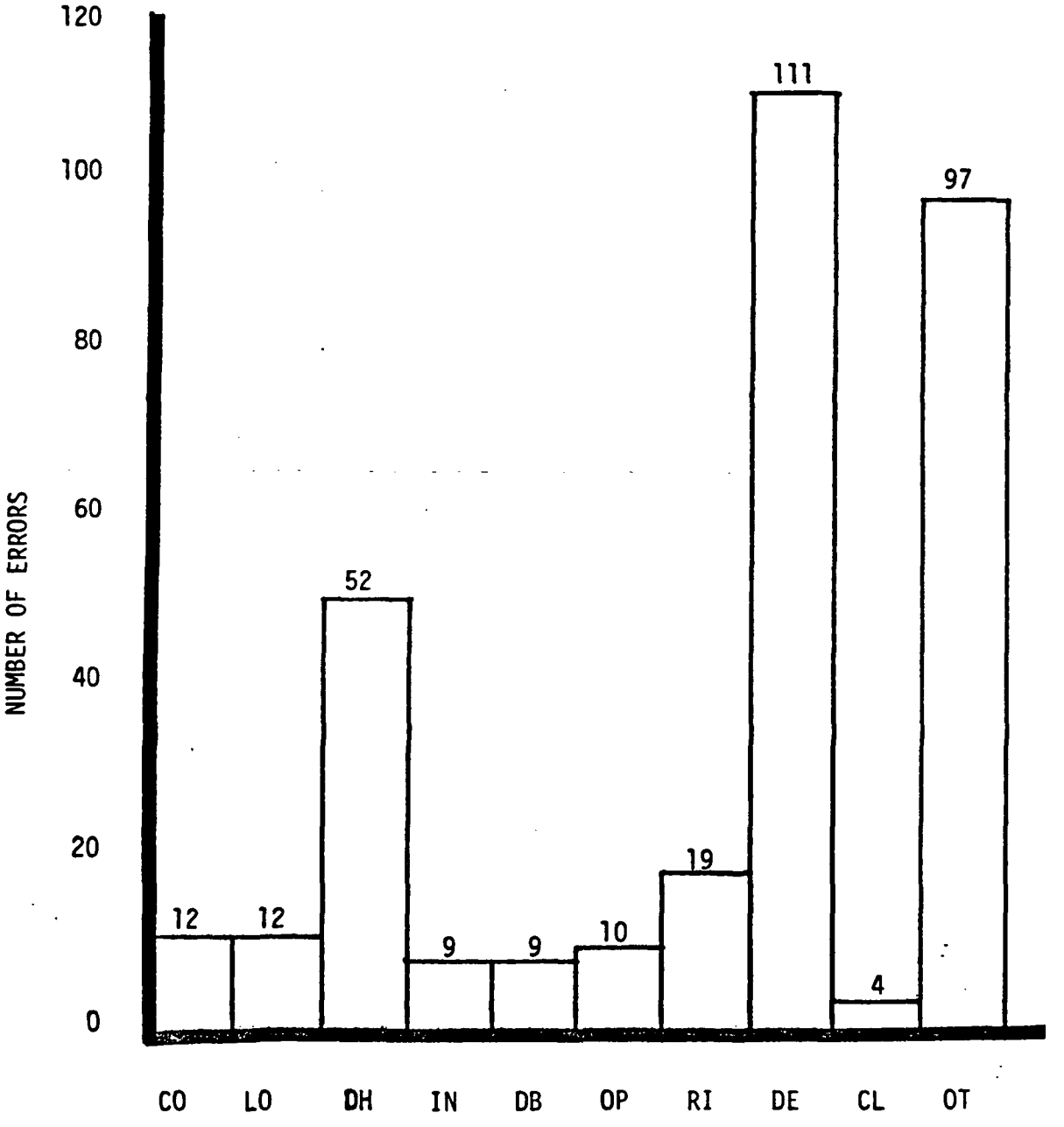


FIGURE 6-10

LEVEL A CRITICALITY BY ERROR CATEGORY



or understand operational anomalies and categorized them as critical to get immediate attention. Again, this data emphasizes the need to revamp the existing anomaly reporting procedure and to investigate ways of improving the man/machine interface.

6.2.11 Level B Criticality by Error Category

This histogram, Figure 6-11, was produced by counting all the level B (dangerous) errors for each of the ten error categories. The histogram identifies which of the error categories caused which proportion of incidences. The preliminary observations we can make based on this histogram are as follows:

- "Other" errors were the most common cause of dangerous errors. Again, we can attribute this to the user who wants attention and misunderstands the system. Also, hardware errors that were misclassified distorted the counts.
- Design errors were again another prevalent cause of dangerous errors.
- Data handling errors were again a major cause of dangerous errors.

6.2.12 Level C Criticality by Error Category

This histogram, Figure 6-12, was produced by counting all the level B (minor) errors for each of the ten error categories. The histogram identifies which of the error categories caused which proportion of level B error incidences. The preliminary observations we can make based on this histogram are as follows:

- "Other" errors were again the most common cause of minor errors.
- Design errors were again a prevalent cause of minor errors.
- Data handling errors were again a common cause of minor errors.

The data confirms the trends experienced under all three severity by error type counts. Some approach to rectify the problems identified seems in order.

6.2.13 Subsystem CMF Errors by Software Revision by Criticality

RCI researchers felt that the five most error-prone subsystems should be subjected to further analysis. We felt that interesting trends could be

FIGURE 6-11
LEVEL B - CRITICALITY BY ERROR CATEGORY

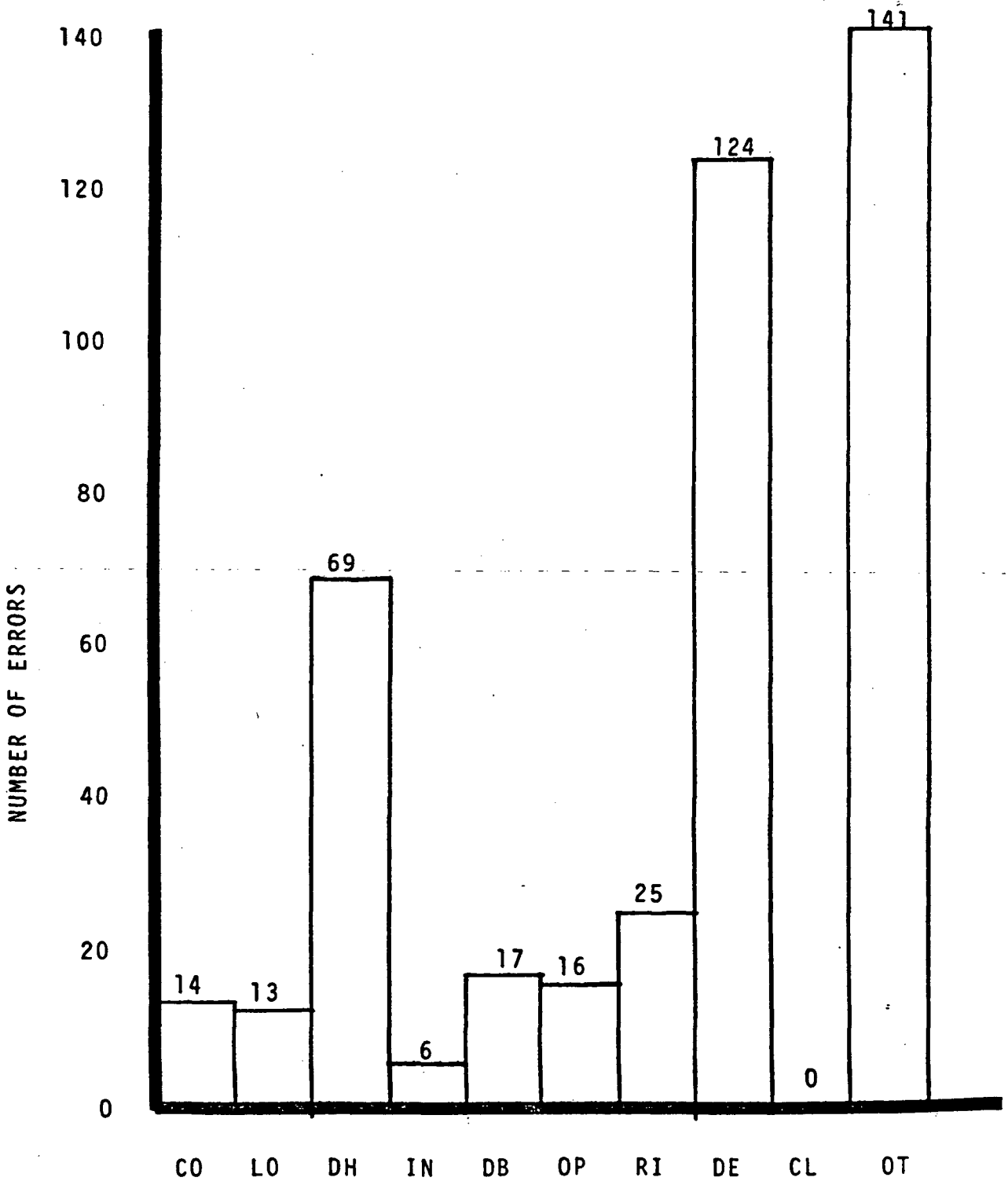
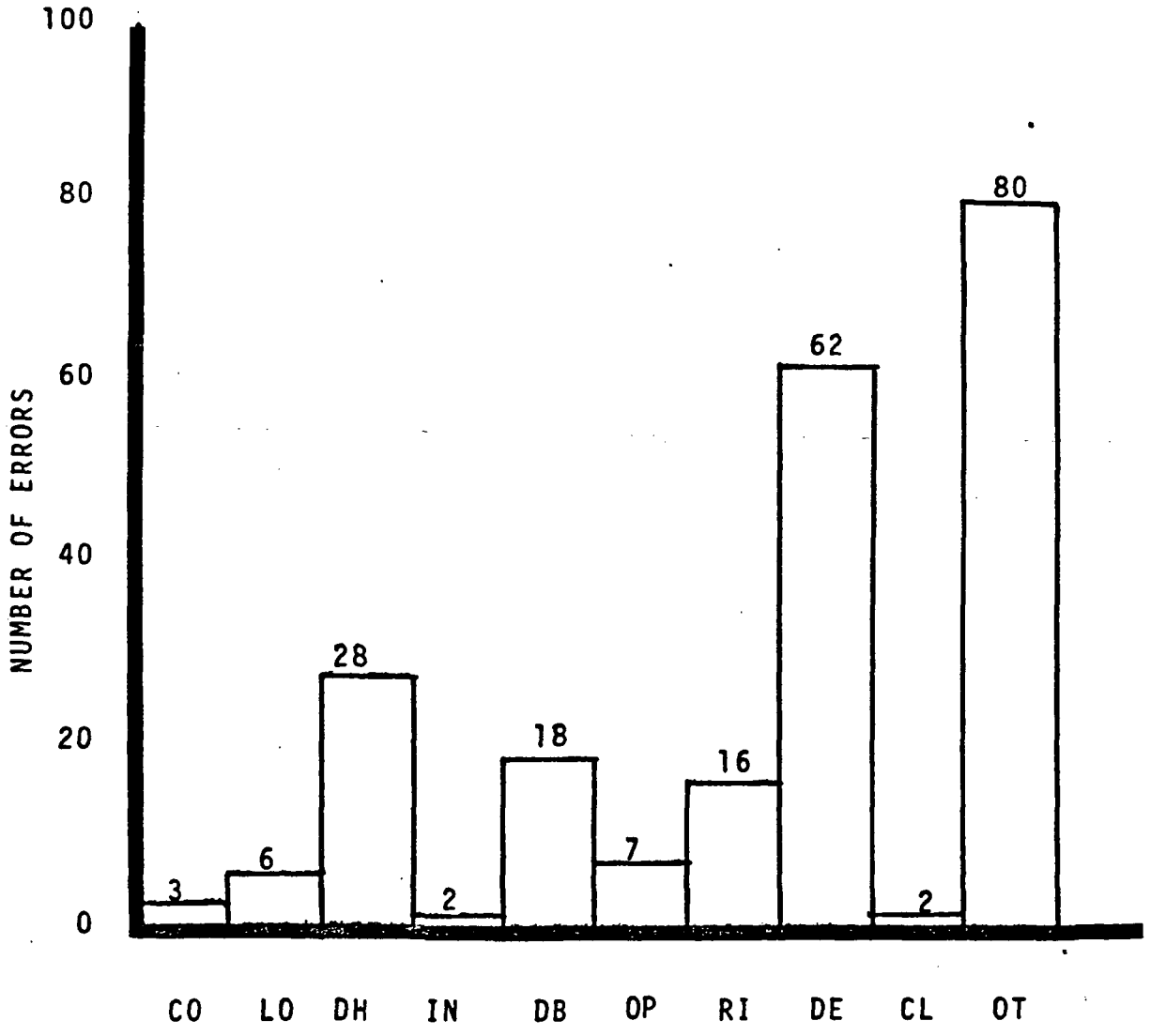


FIGURE 6-12

LEVEL C - CRITICALITY BY ERROR CATEGORY



identified by looking at errors by revision by criticality. We then reduced the data and included them in this report in this and the next four subparagraphs.

This histogram, Figure 6-13, was produced by counting all the errors experienced for subsystem CMF and separating them under the various software revisions and error criticality levels. The first five releases including the initial issuance plus revisions A through D of the subsystem were plotted. The histogram shows the number of errors in each criticality level for each software revision. The preliminary observations we can make based on this histogram are as follows:

- As expected, the number of errors associated with subsystem CMF decreased as a function of time and experience. What was not expected was the extremely high number of level A and B errors during release D. This revision had more level A and B severity errors than any other release with the exception of the initial issuance of the subsystem. There could be several reasons for this phenomenon. For example, release D might have included major enhancements to functional capabilities that caused considerable rework. Yet, such a large number of severe errors should not be present so late in a program's life especially when there is so much experience available with it. RCI believes some further investigation is warranted because this trend defies logic and may be significant.
- The decrease in level C errors to a steady state was expected. Level A and B error did not exhibit such a trend indicating possible problems in the initial release of the subsystem. The acceptance testing procedures used therefore seem suspect and should be reviewed.

6.2.14 Subsystem CPA Errors by Software Revision by Criticality

This histogram, Figure 6-14, was produced by counting all the errors for subsystem CPA and seperating them under various software revisions and error criticality levels. The initial release and revisions A through D were included since they all had significant error histories. The histogram shows the number of errors in each criticality for each software

FIGURE 6-13

SUBSYSTEM CMF - ERRORS BY SOFTWARE REVISION BY CRITICALITY

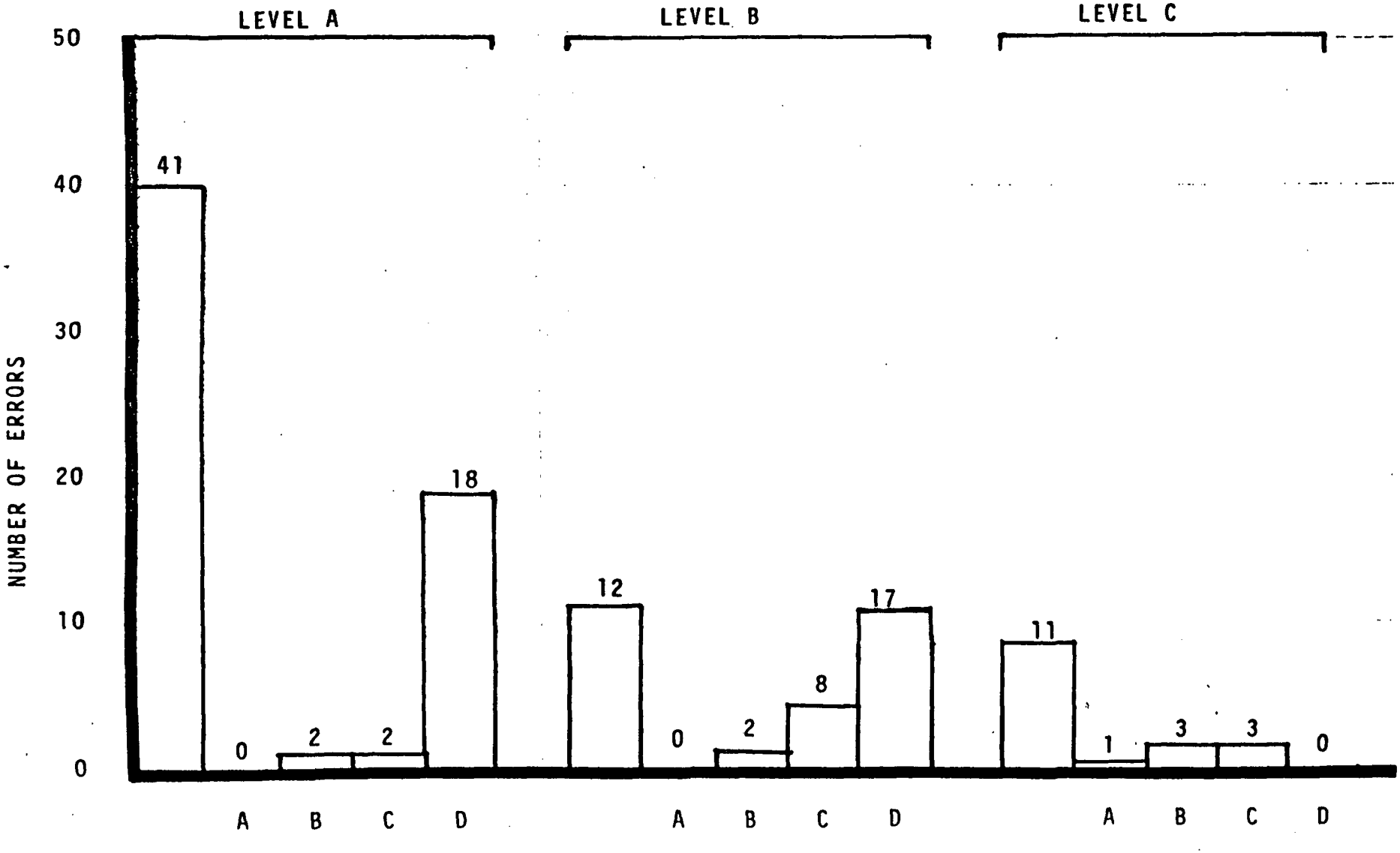
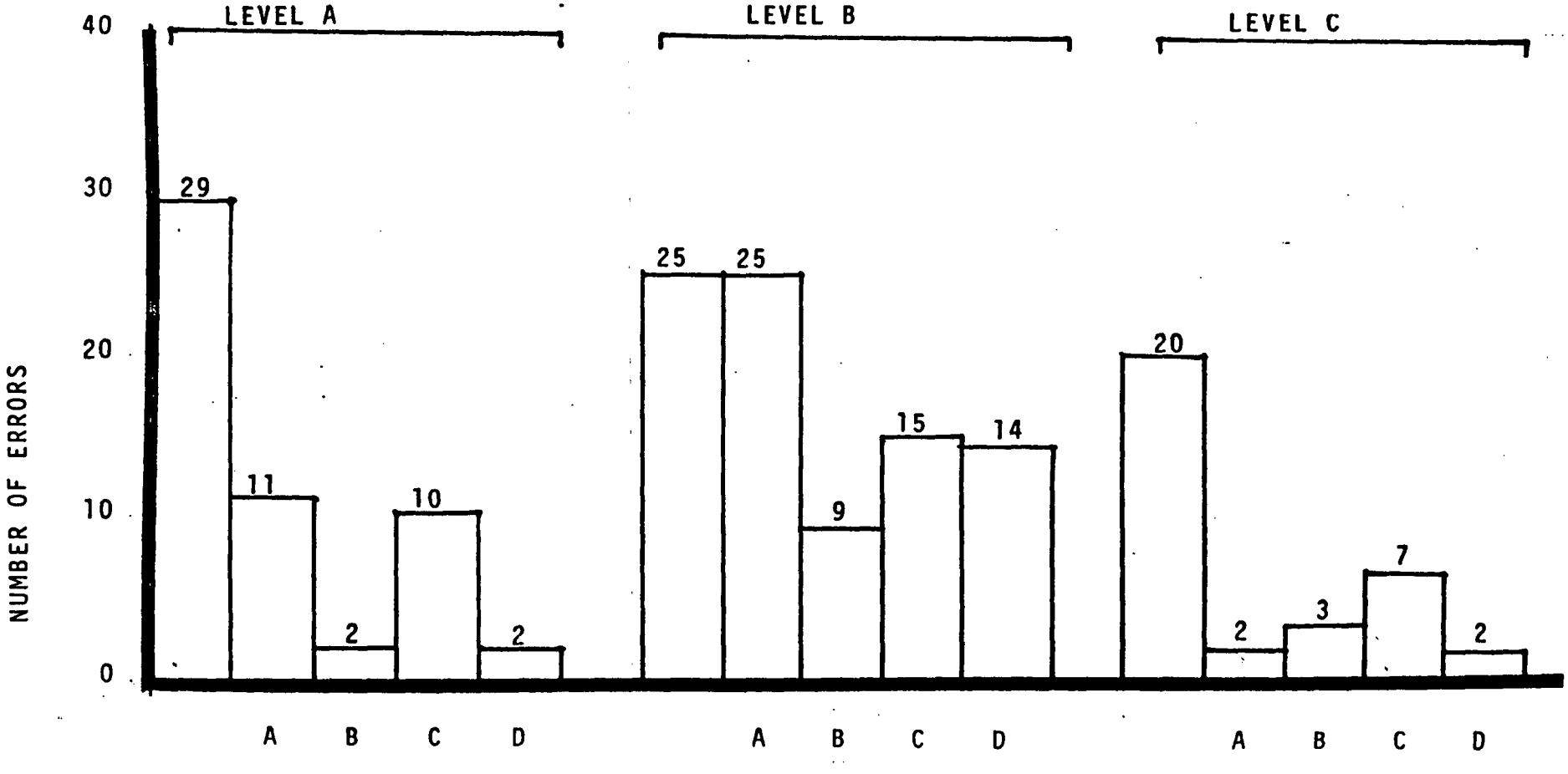


FIGURE 6-14
SUBSYSTEM CPA - ERRORS BY SOFTWARE REVISION BY CRITICALITY



revision. The preliminary observations we can make based on this histogram are as follows:

- The trend was toward lower error incidence during most software revisions with all error severity categories nearly leveling out as revisions were made. This trend is what was expected and demonstrates that the history learned as a function of time was being factored into the subsystem.
- Level B error maturing did not accelerate as rapidly as did levels A and C. Some review might be warranted to understand this phenomenon more fully.

6.2.15 Subsystem DSS Errors by Software Revision by Criticality

This histogram, Figure 6-15, was produced by counting all the errors for subsystem DSS and seperating them into various software revisions and error criticality levels. The initial software release and revisions A through B were plotted because they were the only revisions for which we had data. The histogram shows the number of errors of each criticality level for each software revision. The preliminary observations we can make based on this histogram are similar to those stated for subsystem CPA (Paragraph 6.2.14) and would be redundant to state again.

6.2.16 Subsystem DST Errors by Software Revision by Criticality

This histogram, Figure 6-16, was produced by counting all the errors for subsystem DST and separating them into various software revisions and error criticality levels. The initial software release and revisions A through C were plotted because the data for them were available. The histogram shows the number of errors of each criticality level for each software revision. The preliminary observations we can make based upon this histogram are as follows:

- There was a general decline in error incidence until release C. Possibly, the subsystem became tricky to modify at that time due to improper enforcement of standards or high difficulty. Again, the trend is counter-intuitive and should be investigated.
- There was a trend towards a steady state until release C came along. Again, investigation might be warranted to gain insight into the cause for this phenomenon.

FIGURE 6-15
SUBSYSTEM DSS - ERRORS BY SOFTWARE REVISION BY CRITICALITY

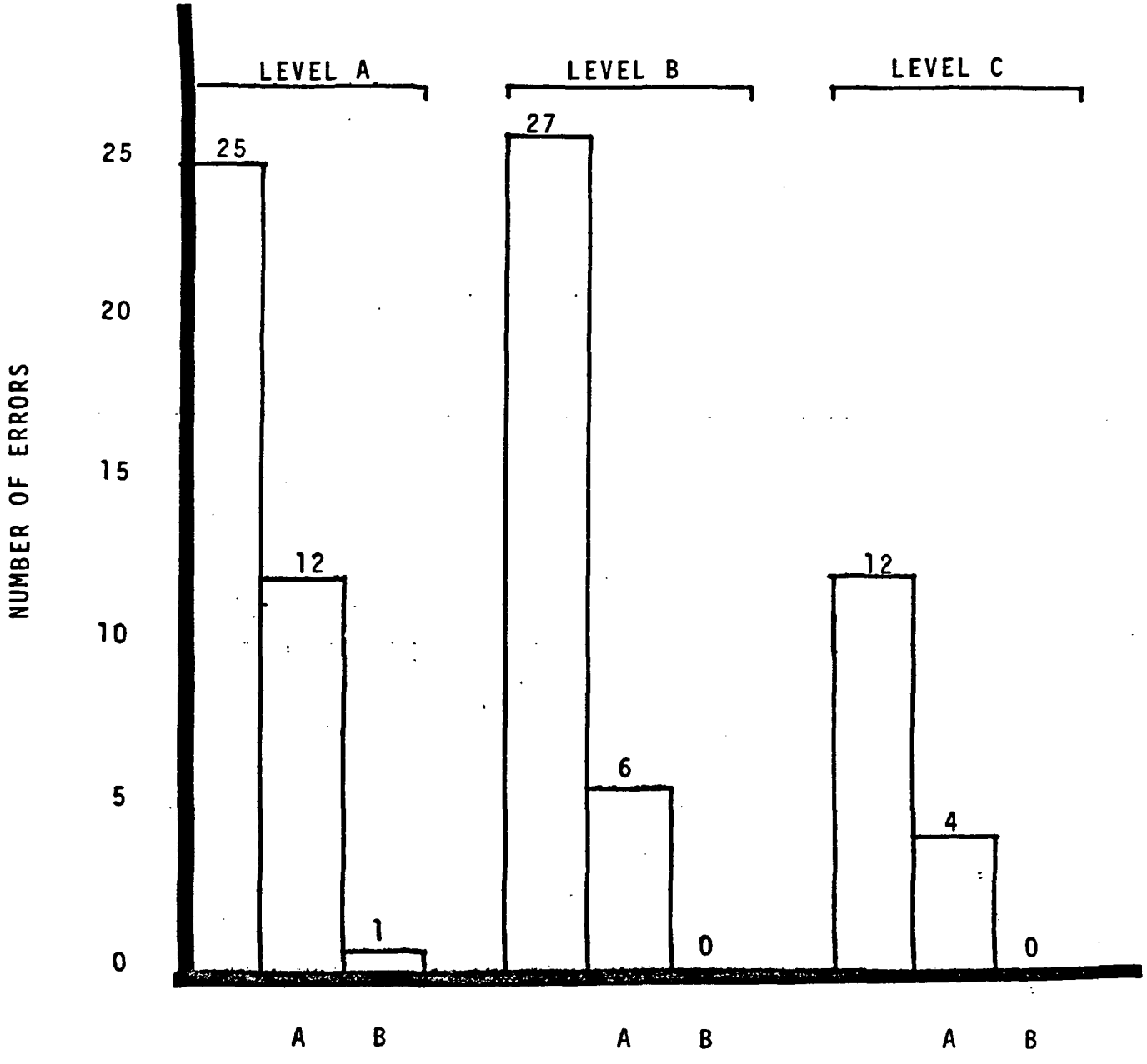
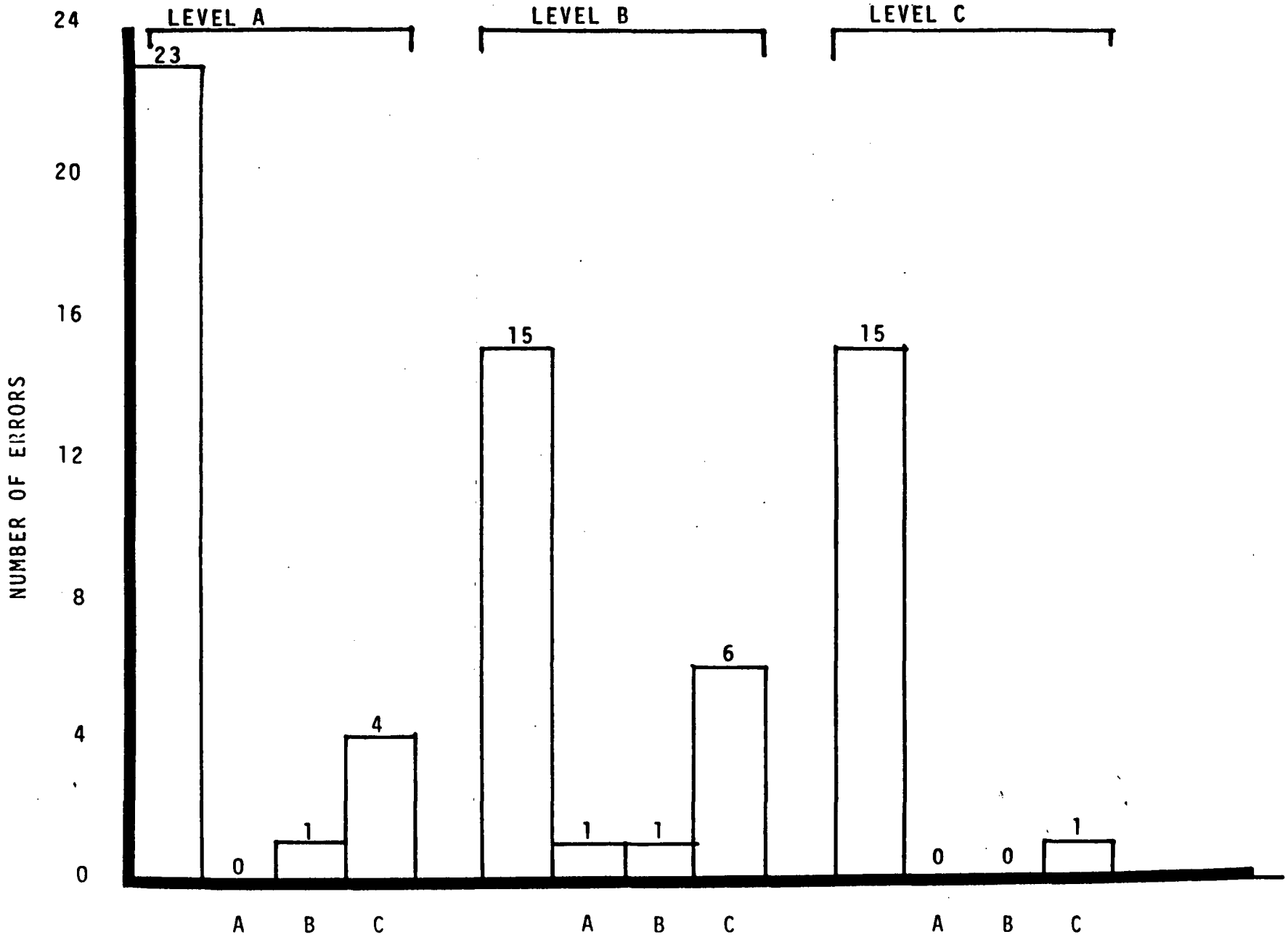


FIGURE 6-16
SUBSYSTEM DST - ERRORS BY SOFTWARE REVISION BY CRITICALITY



6.2.17 Subsystem NTK Errors by Software Revision by Criticality

This histogram, Figure 6-17, was produced by counting all the errors for subsystem NTK and seperating them into various software revisions and error criticality levels. The initial software release and revisions A through E were plotted since all had significant error rates. The histogram shows the number of errors for each criticality level for each software revision. The preliminary observations we can make based on this histogram are as follows:

- The number of level A errors decreased consistently by release basis illustrating that learning was feedback to eliminate critical errors in this subsystem. Possibly, the procedures used for this subsystem could be employed by others to counter the trends we observed that were counter intuitive.
- The numbers of errors in each level seem to reach a steady state solution. This trend compares well with what others have observed on similar data collection activities and illustrates the maturing process that was expected.

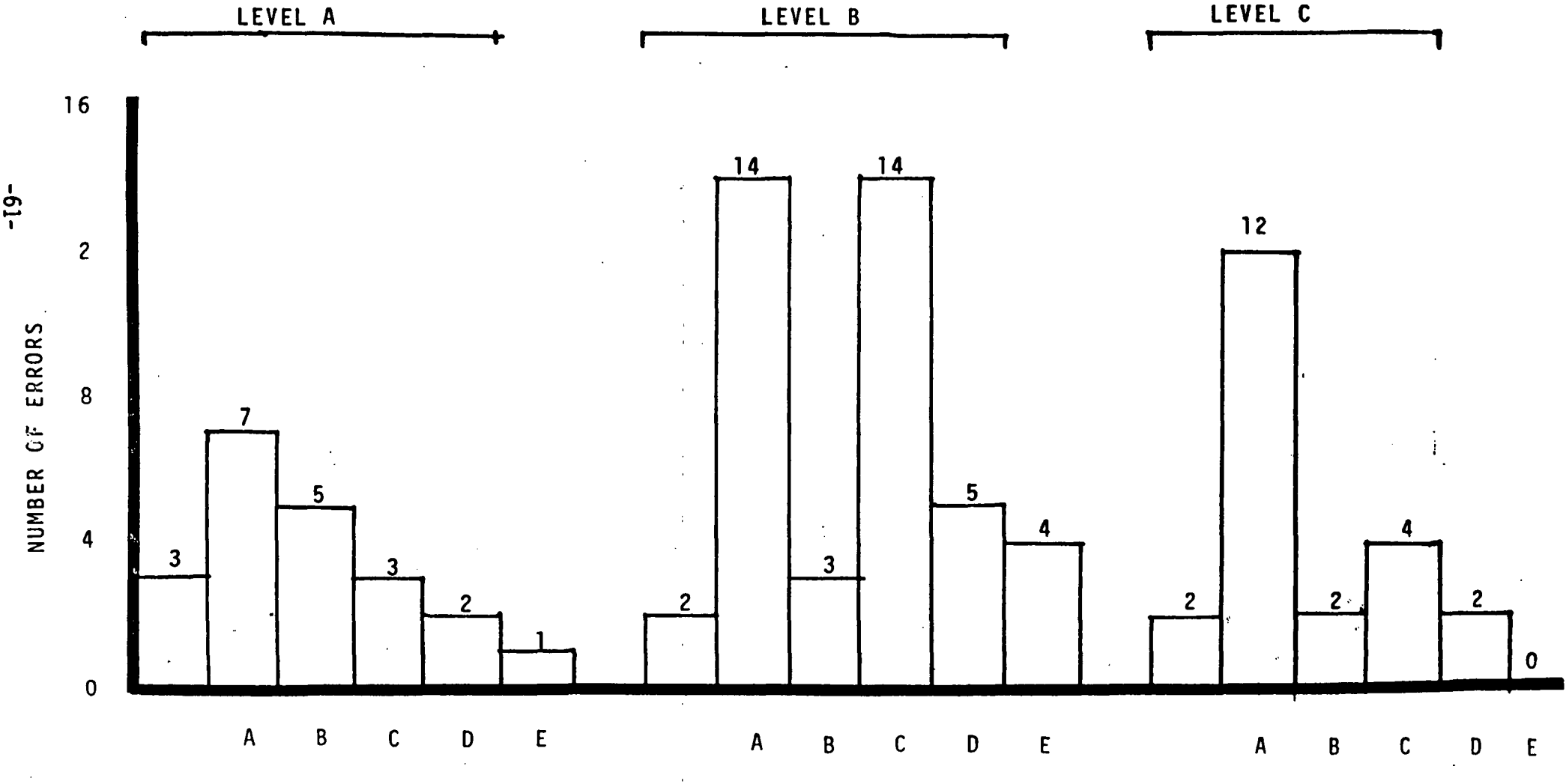
6.3 Comparisons With Other Software Error Studies

As we mentioned in our introduction, many error studies have been conducted in the past. These serve as useful benchmarks for DSN when it comes to understanding what their data means. The purpose of this paragraph is therefore to improve understanding by comparing the DSN/Mark 3 error data to data revealed by other error studies. The two studies chosen for our comparative analysis were done by Logicon for the United States Army (Reference 2) and The Aerospace Corporation (Reference 4) for NASA Langsley Research Center (LaRC).

The NASA/LaRC software error data was compiled during the development phase of data base software for a multi-sensor tracking system (LSDB). This study was chosen for comparison because the software development phase of DSN/Mark 3 provided little data for our researchers. We wanted to see if we could shed some light on the nature of errors that occurred during the development process using this study.

The Logicon software error data was compiled from eleven seperate

FIGURE 6-17
SUBSYSTEM NTK - ERRORS BY SOFTWARE REVISION BY CRITICALITY



Independent Verification and Validation projects done for the United States Army. These projects varied in type (included tactical and command and control) and size of software produced. This study was chosen as a benchmark because it encompassed data that was taken from software that had been transitioned into operations by the developing organization. This data is also somewhat similar to the DSN/Mark 3 "acceptance" data.

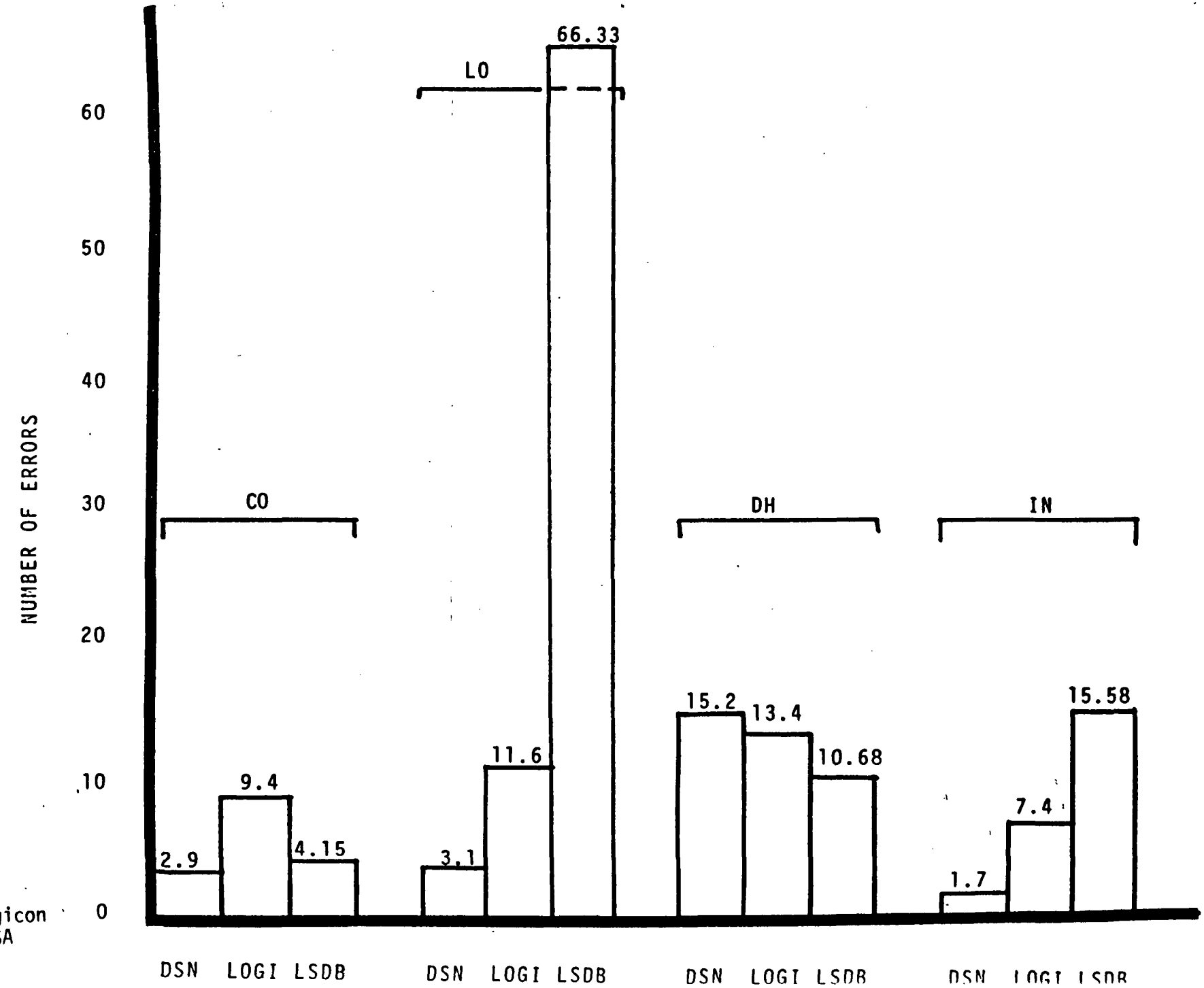
As our first step in comparison, the error categories used in the Logicon and NASA/LaRC studies were mapped into the DSN/RCI taxonomy as shown in Figure 3-1. Next, a comparison histogram was generated as Figure 6-18. Only those categories containing pertinent information were plotted in the comparison graph. Finally, an analysis of what the data meant was conducted. All error categories were compared using percentages of errors so that the data could be normalized.

The following observations can be made from this comparing of data:

- DSN/Mark 3 had a smaller percentage of computation errors when compared against the other two data bases. This was expected because of the nature of the DSN/Mark 3 software.
- DSN/Mark 3 has a smaller percentage of logic errors when compared against the other two data bases. The extremely large percentage of logic errors reported by the LSDB data was expected because these errors were reported during initial development. Similar percentages of logic errors probably occured during DSN/Mark 3 development, but were not reported. This data indicates that DSN has had some success in removing logic errors early using its existing practices. Such practices should be continued and possibly should be enhanced as a result.
- Data handling errors were a large percentage of all three data bases. This data indicates that there is a need for better standards and enforcement measures for correctly handling data items in most large-scale, real-time software projects.
- DSN/Mark 3 has a smaller percentage of interface errors when compared against the other two data bases. This demonstrates that DSN's approach to modularizing subsystems and checking in detail the sequencing and control logic seems to be working. Because these

FIGURE 6-18

COMPARISON OF DSN/MARK 3 DATA
WITH LOGICON AND NASA/LaRC DATA BASES



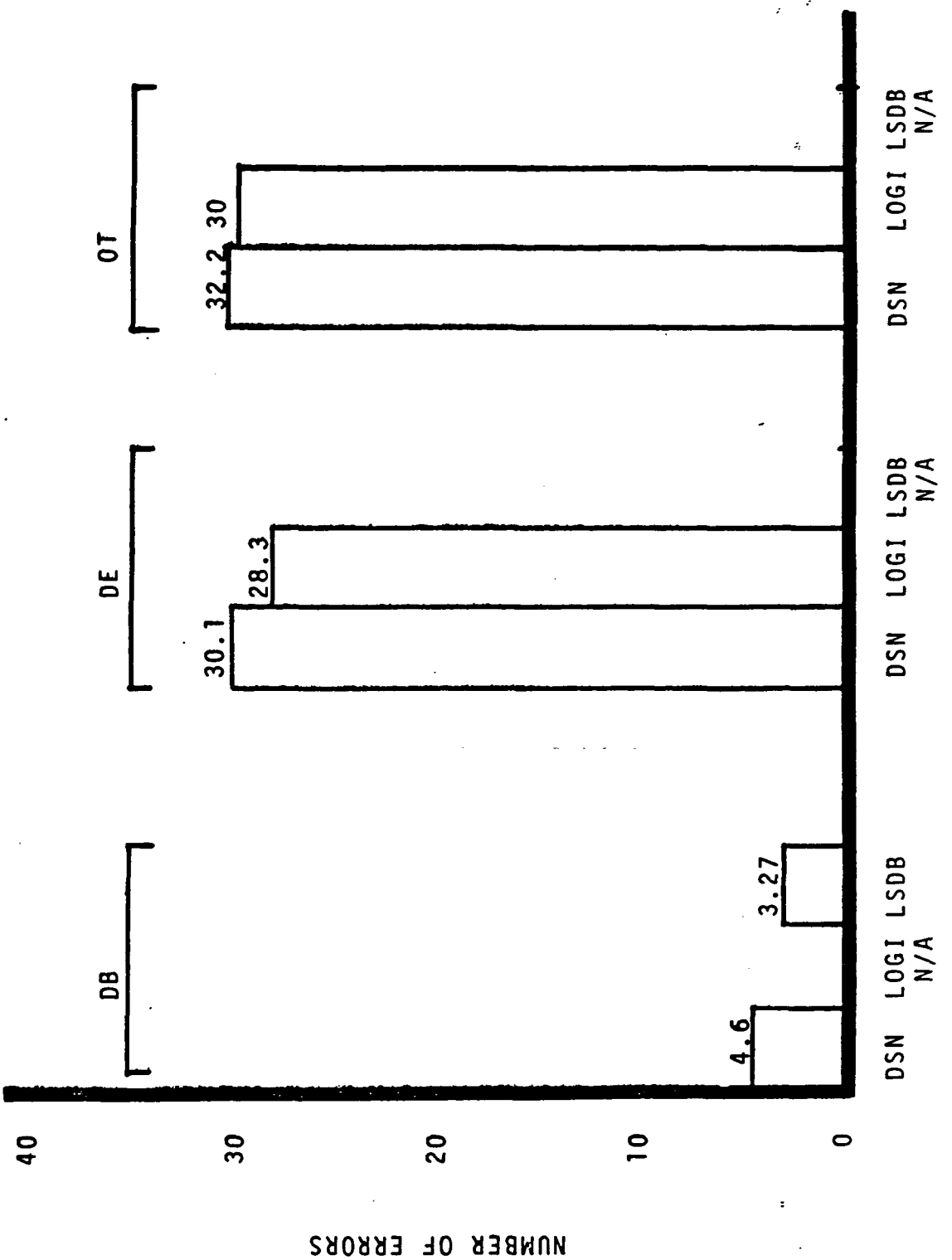


FIGURE 6-18 (cont'd.)

were critical errors in the other studies, the DSN efforts should be continued.

- Data base errors percentages were similar between DSN and LSDB. Logicon did not categorize data base errors as a main classification so further breakdown of the data is currently impossible.
- Design errors percentages exhibited similar percentages for both the DSN/Mark 3 and Logicon data bases. This again validates the need for better design review procedures to be established early in software development. It also reenforces the need for better quality control and for automated tools.
- Other errors also exhibited high correlation between DSN/Mark 3 and Logicon data bases. Although Logicon has no "other" errors per se, classifications such as violation of programming practices and documentation errors were comparable with DSN "other" errors. Again, this data indicates that there is a need to improve current procedures for anomaly reporting and to make sure people understand how to use them.

This Page Intentionally Blank

SECTION 7

PHASE 2 STUDY PLAN

The purpose of this section is to outline a plan of action for Phase 2 of the project. Also included are several recommendations for studies which could be of benefit to the DSN.

7.1 Goals of Phase 2

Phase 2 of this study will be conducted to accomplish the following goals:

- To identify and statistically validate DSN/Mark 3 software error trends using the software error data base developed during Phase 1 as a basis.
- To enhance the DSN/RCI software error taxonomy so that it can be applied to collect meaningful data for the DSN/Mark 4 project.
- To define enhanced anomaly reporting forms and procedures for use now and in the future by DSN personnel.

7.2 Summary of Recommendations

Our action plan is summarized in Table 7-1. Recommendations are identified within this table to correct problems found as a result of our Phase 1 analysis. These recommendations extend Phase 2 to encompass a number of optional studies each of which could be pursued to benefit the DSN.

7.3 Statistical and Trend Analysis

The first task planned for Phase 2 is Statistical and Trend Analysis. The preliminary review of the DSN/Mark 3 software error data base seems to identify error tendencies in several areas. However, the trends identified by this preliminary evaluation neither have been statistically validated nor statistically analyzed. Additional data base analysis is needed to both validate the trends that have been suggested by the Phase 1 data (see histograms in Section 6) and identify other error tendencies that were not apparent when viewing only the raw data. The statistical and trend analysis recommended by the study team for conduct during Phase 2 could pave the way for development of checklists, improved procedures and automated tools all of which could be used to improve quality through error reduction.

7.4 Extended DSN/RCI Anomaly Reporting Procedures

The next task planned for Phase 2 is the enforcement of anomaly reporting procedures. The objective of the DSN/RCI error taxonomy developed during Phase 1 of this study was to create a meaningful software error data base for the DSN using existing anomaly information. Application of the DSN/RCI taxonomy to the existing anomaly reports pointed out the need to collect additional data during the software life cycle. The purpose of this task will be to extend the error taxonomy to encompass the additional desired error data. In addition, the task would generate a new anomaly reporting form and procedures for completing it.

This task will also provide procedures for "request for discussion/review". This request procedure will help eliminate the use of the anomaly reporting form as a vehicle for requesting design changes, will improve communications and will provide more consistant anomaly data.

These forms can then be the basis of a management error tracking system, provide an up to date error data base and assure consistent anomaly reporting during the DSN/Mark 4 project.

7.5 Additional Studies

7.5.1 Examine Man/Machine Interfaces

One of the items made apparent by our analysis of the DSN/Mark 3 error data base was that many errors were caused by incorrect man/machine actions. Not only were these errors identified by the "operations" category but some were included within the "other" category. Historically, many of the operator errors were the result of unclear or overly complicated operator procedures. The "other" errors seem to result from similar causes. Yet, these resulted when programmers were interfacing with the machine.

We recommend further investigation in this area. This investigation will examine errors caused by both operators and programmers when they work with the operational software. It would attempt to define methods of simplifying man/machine interfaces and making DSN software more error tolerant. This study could help eliminate a significant portion of the errors that occur once the software package is operational in a production environment.

TABLE 7-1
ACTION PLAN SUMMARY

Finding	Recommendation
• DSN could use error data as a management tool.	• Extend DSN/RCI taxonomy to collect data. • Automatically collect and reduce error data. • Quantify software reliability using existing metrics. • Develop statistical method for determination of when testing is complete.
• The anomaly reporting process should be more tightly controlled.	• Enhance the error reporting form and make someone responsible for data validation.
• Requirements and design phase products should be more carefully tracked and reviewed.	• Examine automated tools for requirements and design analysis. • Investigate use of IV&V organization.
• Improved interface and reporting procedures could potentially eliminate many anomalies.	• Investigate improving man/machine interfaces. • Develop improved anomaly reporting systems and procedures.
• Apparent trends in the error data base must be validated.	• Perform detailed statistical and trend analysis of the software error data base. • Perform subfactor (subclassification) trend analysis to identify secondary error effects.

7.5.2 Automated Tools Use

Software tools can serve as powerful aids in the design, development, test, operations and maintenance of computer software. They can assist the analyst, manager, programmer and user by providing meaningful information that can be used to automate parts of the software effort and validation process thereby increasing software reliability and productivity. Some of these tools can validate requirements specifications, design specifications and program code. Others can identify highly complex program modules so these may be tested more thoroughly. This study will explore application of existing automated tools during the development of software for the DSN/Mark 4 system. These tools would be used to reduce the DSN error rate by early detection and correction of errors. Use of tools could provide significant error reduction and cost savings to JPL.

7.5.3 Detailed Sub-Classification of Error Categories

The DSN/RCI error taxonomy used to create the error data base categorized error types (such as computation, logic and data handling) into ten major error classifications. This was done to provide meaningful data for statistical and trend analysis and to avoid the syndrome of confusing classification schemes. However, more detailed sub-classification of errors from the three most error-prone categories (requirements, design and other) can yield additional information about the nature of these errors. This could allow us to more precisely identify potential areas of concern during DSN/Mark 4 development. This sub-classification would be performed by re-examining the anomaly reports for significant trends in error categories. For example, design errors may be broken into design errors caused by misinterpretation of requirements, by inadequate documentation, by not clearly thinking through the problem, and other significant design error sub-classifications. This additional sub-classification could highlight areas where better techniques for design and requirements review could be profitably used during the DSN/Mark 4 development.

7.5.4 Statistical Determination of When Testing is Complete

During the development of the DSN/Mark 4 system software, testing will be a major concern because of its high costs and the potential deleterious impact of undetected software errors. This study will address the testing problem by using statistical techniques and Halstead's Software Science to

help set testing objectives. These techniques would also be used to determine whether or not objectives have been met. Included in this study would be an assessment of state-of-the-art methodologies for estimating the number of software errors within a system, use of the Graeco Latin Square statistical technique for generating the minimum test data sets needed to detect errors and recommendations for automating and applying these techniques during DSN/Mark 4 testing. This study has the potential to answer the question faced by DSN that asks "How much testing is enough for large, distributed systems?".

7.5.5 Software Reliability Measurement

Software reliability is often talked about, but seldom quantified in any useful manner. Using state-of-the-art statistical methods, this study will answer the question "How reliable is the software?". Existing DSN/Mark 3 computer utilization records will be used to quantify DSN/Mark 3 reliability over the life cycle using the Musa model (Reference 8). Additionally for DSN/Mark 4, computer utilization forms will be evaluated and modified if necessary to assure that reliability can be accurately measured. This quantification will consist of a mean time between failure (MTBF) analysis of the software package. Since failure is an often misconstrued term, a precise definition of failure will be established in terms of DSN performance. This quantification of reliability will allow for better management tracking of system performance, better means of system evaluation and performance measurement and earlier evaluation of the effects of system modification on system performance.

This Page Intentionally Blank

REFERENCES

(1) N.E. Willnorth, M.C. Finfer and M.P. Templeton, System Development Corporation, Software Data Collection Study Summary and Conclusions, RADC-TR-76-329, Volume I, December 1976.

(2) Logicon, Inc., Verification and Validation for Terminal Defense Program Software, Report No. HR-74012, 31 May 1974.

(3) H.E. Williams, Jr., T.A. James, A.A. Beaureguard and P. Hilcoff, Raytheon Company, Software Systems Reliability: A Raytheon Project History, RADC-TR-77-188, June 1977.

(4) H. Hecht, W.A. Strum and S. Trattner, The Aerospace Corporation, Reliability Measurement During Software Development, NASA-CR-145205, September 1977.

(5) D.D. Galorath and D.J. Reifer, Software Error Taxonomy Annotated Bibliography, RCI-TR-002, 31 October 1980.

(6) D.J. Reifer and D.D. Galorath, Recommended DSN Software Error Classification Taxonomy, RCI-TN-001, 31 October 1980.

(7) C. Johnsen, Anomaly Reporting Procedure for Software Anomaly Report Form A Section 338, March 1980.

(8) J.D. Musa, Software Reliability Measurement, Bell Telephone Laboratories, 1977.

This Page Intentionally Blank

ATTACHMENT A

ANNOTATED BIBLIOGRAPHY

Reifer Consultants, Inc.

2733 Pacific Coast Highway, Suite 203 • Torrance, California 90505

PREFACE

This document reports the results of a literature search conducted to identify source material useful in our developing a software error classification scheme for the Deep Space Network's Mark 3 System.

Part I summarizes five major taxonomies. Each is abstracted and evaluated in terms of its strengths and weaknesses. Part II provides an annotated bibliography of other documents which influenced our recommended error taxonomy. Part III lists other references used during the investigation.

PART I

MAJOR TAXONOMIES

HUGHES-FULLERTON

Bowen, J.B., Standard Error Classification to Support Software Reliability Assessment, Proceeding 1980 National Computer Conference, 1980, p.697-705.

This article describes the Hughes-Fullerton error classification scheme. The methodology was developed for internal use, during all software development phases. Seven major error classifications (Based on TRW's major categories) have been defined. Appropriate sub-classifications are added as needed for a particular project.

As errors are reported, the software development phase in which the error was originally introduced into the system is recorded.

Three error severity classifications track the impact of the error, critical, major, and minor. Trivial errors are ignored.

STRENGTHS

1. Error severity is captured.
2. The development phase where the error originally occurred is identified.
3. There are few enough error categories that a person can remember them easily.
4. The assignment of error sub-categories for a particular project helps assure consistant data recording of unique errors.

WEAKNESSES

1. The more sub-categories used the more difficult the classification process becomes.

LOGICON

Dana, J.A. and Blizzard, J.D., Verification and Validation for Terminal Defense Program Software, The Development of a Software Error Theory to Classify and Detect Software Errors, Logicon Inc., for System Development Corporation, HR-74012, May 1974.

This study defined twelve major error categories, each with several sub-categories. The major categories are well enough defined to encompass errors from commercial, scientific, real-time systems, application or system software. In addition to the definition of the errors, error detection methods are recommended.

Four error severity categories (catastrophic, serious, moderate and trivial) have been defined.

The major consideration of this error report was the validation phase rather than program development.

STRENGTHS

1. Error severity is captured.
2. The large number of error sub-categories assures consistant error recording if personnel apply them correctly.

WEAKNESSES

1. The development phase where the error occurred is not identified.
2. The large number of error sub-categories make the system difficult to use.

NASA/GSFC

Forms and Instructions Change Report Form, NASA Goddard Space Flight Center, June 1978.

Eight major error categories are defined in this two page form, with two major sub-classifications for design and implementation errors. The development phase (requirements, functional specs, design, coding and test, other, and can't tell) is also captured on the form.

Errors that were caused by previous changes are noted since this is often a significant percentage.

The activites used to validate the program, detect the error and find its cause are defined in a matrix so the person filling out the form can quickly check off those actions taken. This matrix shows such things as programmer interaction in addition to normal test runs.

STRENGTHS

1. The development phase where the error originally occurred is identified.
2. The methodologies for error location and correction are captured using a simple matrix.
3. There are few error categories and these are merely checked off on the form.

WEAKNESSES

1. Error severity is not specifically recorded.

NASA/LaRC

Hecht, H., Sturm W.A. And Trattner S., Final Report, Reliability Measurement During Software Development, The Aerospace Corporation, for Langley Research Center, NASA Contractor Report 145205, Sept. 1977.

This study defined twelve error categories with several examples of errors within each category. This allowed the use of a simple one page failure analysis report form.

Five error severity categories were used: system crash, dependent job failure, local job failure only, real time failure and other.

The error categories are based on TRW's major error classifications.

STRENGTHS

1. Error severity is captured.
2. There are few enough error categories that a person can remember them easily.

WEAKNESSES

1. The development phase where the error originally occurred is not recorded.

TRW

Thayer, T.A., Lipow, M., Nelson, E.C., Software Reliability Study, TRW Defense and Space Systems Group, Mar. 1976.

The TRW study defined twelve error categories with numerous sub-categories. First, as errors occurred they were used to define categories and sub-categories. This method had a tendency to yield a new sub-category for each error. Then, a reduced set of sub-categories was generated through analysis of the original sub-categories. The development phase in which the error occurred, if known was also tracked.

Although the study did not track error severity, the final recommendations contained five severity classifications for future studies.

The TRW study results were used as the basis of many future error classification studies.

STRENGTHS

1. The development phase where the error originally occurred is identified.
2. The large number of error sub-categories assures consistant error recording if personnel apply them correctly.

WEAKNESSES

1. Although the study recommends the inclusion of severity information, it was not considered during the study.
2. The large number of error sub-categories makes the system difficult to use.

PART II

ANNOTATED BIBLIOGRAPHY

1. Chief of Naval Materiel, Military Standard for Weapon System Software Development ,MIL-STD-1679 (Navy) AMSC No. 23033, Dec. 1978.

This specification defines methodologies for producing accuracy and effective operation of software, methods of implementing software changes, and factors that will reduce life cycle costs. Precise definitions of various software error types and severity are given.

2. Final Technical Report, Software Data Collection Study Summary and Conclusions, System Development Corporation, RADC-TR-76-329, Jun. 1976.

This study describes the methodology used to create a software data base including costing, performance and error data. Methods of classifying error type and when the error was generated are given.

3. Final Technical Report, Software Systems Reliability: A Raytheon Project History, Raytheon Company, RADC-TR-77-188, Nov. 1976.

This study describes the methodology used to create a software error data base. The error classification scheme is an extended version of the TRW error classifications.

4. Final Technical Report, Software Error Data Acquisition, Boeing Aerospace Company, RADC-TR-77-130, Apr. 1977.

This study describes the methodology used to create a software error database and the results of analysis of the error data. Errors were categorized using the TRW error classifications. Boeings results were compared to published TRW results.

5. Final Technical Report, Software Data Collection and Analysis: A Real-Time System Project History, IBM Corporation, RADC-TR-77-192, Jun. 1977.

This study describes the methodology used to create a software error data base and describes the results of the error analysis. The error classification scheme is an extended version of the TRW error classifications.

6. Weiss D.M., Evaluating Software Development by Error Analysis: The Data From the Architecture Research Facility, Naval Research Laboratory Communications Sciences Division, NRL RPT 8268, Dec. 1978.

This study describes a technique for collecting data on software errors. An earlier version of the NASA/GSFC change report form was utilized. Steps used to evaluate a methodology of error analysis were also described.

This Page Intentionally Blank

PART III

REFERENCES

1. Proceedings of Second Summer Software Engineering Workshop, Software Engineering Laboratory, Goddard Space Flight Center, Sept. 1977.

2. Proceedings From the Third Summer Software Engineering Workshop, Goddard Space Flight Center, Sept. 1978.

3. Proceedings From the Fourth Summer Software Engineering Workshop, Goddard Space Flight Center, Nov. 1979.

4. Proceedings: International Conference on Reliable Software, Apr. 1975.